

IMT School for Advanced Studies, Lucca
Lucca, Italy

**A Unified Geometric Deep Learning Framework for
Financial Time Series: Integrating Temporal, Graph, and
Signature-Based Representations**

PhD Program in Systems Science
Track in Economics, Networks and Business Analytics
XXXVII Cycle

By
Marco Gregnanin
2026

The dissertation of Marco Gregnanin is approved.

PhD Program Coordinator: Ennio Bilancini, IMT School for Advanced Studies Lucca

Advisor: Prof. Giorgio Gnecco, IMT School for Advanced Studies Lucca

Co-Advisor: Prof. Johannes De Smedt, KU Leuven

Co-Advisor: Prof. Maurizio Parton, University of Chieti-Pescara

The dissertation of Marco Gregnanin has been reviewed by:

Francesco Morandin, University of Pisa

Ming Li, Zhejiang Normal University

IMT School for Advanced Studies Lucca

2026

To my grandmother

Contents

List of Figures	x
List of Tables	xii
Acknowledgements	xv
Vita and Publications	xvii
Abstract	xx
1 Introduction	1
2 Related Works	9
2.1 Financial Time Series Forecasting from Statistical Models to Machine Learning	10
2.2 Graph Neural Networks in Time Series	12
2.3 Signature-Based and Hypergraph-Based Representation Learn- ing for Time Series	16
2.4 Generative Models for Synthetic Financial Time Series . . .	21
2.5 Summary and Research Gap	23
3 Methodology	25
3.1 Graph Theory and Visibility Graph	25
3.2 Graph Neural Networks	29
3.3 Signature Transform	32

4	Signature-Based Network Modeling for Financial Time Series	37
4.1	Introduction	38
4.2	Preliminaries	41
4.2.1	Signature-based Similarity Matrix	42
4.2.2	Filtering Method	44
4.2.3	Community Detection	47
4.2.4	Asset Allocation Strategies	48
4.3	Community Detection	50
4.4	Network Properties	53
4.5	Asset Allocation Framework	58
4.6	Conclusion	62
5	Time Series Forecasting using Geometric Deep Learning	64
5.1	Introduction	65
5.2	Preliminaries	67
5.2.1	Time-Geometric model	67
5.2.2	Statistical Tests	72
5.3	Time Series Forecasting	75
5.3.1	Model Specification	76
5.3.2	Experiments Configuration	78
5.3.3	Numerical Results	80
5.4	Statistical Analysis	83
5.4.1	Problem Formulation	83
5.4.2	Experiments Configuration	84
5.4.3	Numerical Results	85
5.5	Conclusion	89
6	Advances in Geometric Deep Learning	92
6.1	Introduction	93
6.2	Generative Adversarial Networks	96
6.2.1	Proposed Framework	96
6.2.2	Numerical Comparisons	99
6.3	Hypergraph Neural Networks	104
6.3.1	Proposed Framework	105
6.3.2	Numerical Comparisons	113

6.4	Conclusion	117
7	Conclusion	118
A	Signature-Based Similarity	123
A.1	Filtering Method	123
A.2	Asset Allocation Models	123
A.3	Asset Allocation Results	125
B	Time Series Forecasting using Geometric Deep Learning	136
B.1	Evaluation Metrics	136
B.2	Hyperparameters	137
B.2.1	Baseline Models	137
B.2.2	Time-Geometric Model	138
B.3	Empirical Results	140
C	Advances in Geometric Deep Learning	151
C.1	Baseline Models Hyperparameters	151

List of Figures

1	Visibility Graph Algorithm	28
2	Lead-Lag Transformation	33
3	Communities in the Standard and Poor's 500	54
4	Nodes weighted degree with multiple threshold	55
5	Nodes clustering coefficient with multiple threshold	56
6	Comparison between nodes clustering coefficient and nodes weighted degree	57
7	Comparison of stocks standard deviation and nodes degree centrality	58
8	Comparison of stocks standard deviation and nodes eigenvector centrality	59
9	Framework of the proposed model	68
10	Time-Geometric Model Architecture	69
11	Time-Geometric model used to forecast the S&P 500	90
12	Standard and Poor's 500 one-step ahead prediction.	91
13	Standard and Poor's 500 one-step ahead out-of-sample prediction.	91
14	Proposed Generative Adversarial Networks Architecture.	98
15	Sig-Graph GAN (KLD) Cumulative Returns Distribution	102
16	Ablation study for the Sig-Graph GAN	104
17	Proposed Hypergraph framework.	106
18	Proposed Hypergraph Neural Networks Architecture.	109

19	Average metric values for 1 day prediction	142
20	Average metric values for 5 day prediction	144
21	Average metric values for 20 day prediction	146
22	Nemenyi test for 1 day forecasting	148
23	Nemenyi test for 5 day forecasting	149
24	Nemenyi test for 20 day forecasting	150

List of Tables

1	Modularity results and number of clusters discovered . . .	53
2	Standard and Poor's 500 one-step ahead prediction.	81
3	Standard and Poor's 500 one-step ahead out-of-sample pre- diction.	82
4	Pairwise statistical results for one-step ahead prediction . .	87
5	Friedman test	88
6	Generative Adversarial Neural Networks Hyperparameters	101
7	Results for the real and generated data	103
8	Hypergraph Neural Networks Hyperparameters	115
9	Comparison of forecasting models for Hypergraph Neural Networks.	116
10	Critical values of the standard normal distribution.	123
11	Results obtained for the portfolio strategies using the 10 most liquid stocks	126
12	Results obtained for the portfolio strategies using the 20 most liquid stocks	127
13	Results obtained for the portfolio strategies using the 40 most liquid stocks	128
14	Results obtained for the portfolio strategies using the 80 most liquid stocks	129
15	Results obtained for the portfolio strategies with short sell- ing using the 10 most liquid stocks	130

16	Results obtained for the portfolio strategies with short selling using the 20 most liquid stocks	131
17	Results obtained for the portfolio strategies with short selling using the 40 most liquid stocks	132
18	Results obtained for the portfolio strategies with short selling using the 80 most liquid stocks	133
19	Results for the out-of-sample portfolio strategies with and without short selling using the 10 most liquid stocks	134
20	Results for the out-of-sample portfolio strategies with and without short selling using the 20 most liquid stocks	134
21	Results for the out-of-sample portfolio strategies with and without short selling using the 40 most liquid stocks	135
22	Results for the out-of-sample portfolio strategies with and without short selling using the 80 most liquid stocks	135
23	Baseline model's hyperparameter values for one-step ahead prediction	138
24	Baseline model's hyperparameter values for 5-step ahead prediction	138
25	Baseline model's hyperparameter values for 20-step ahead prediction	139
26	Time-Geometric model's hyperparameter values for one-step ahead prediction	139
27	Time-Geometric model's hyperparameter values for 5-step ahead prediction	140
28	Time-Geometric model's hyperparameter values for 20-step ahead prediction	140
29	Average metric values for 1 day prediction	141
30	Average metric values for 5 day prediction	143
31	Average metric values for 20 day prediction	145
32	Pairwise statistical results for 5-step ahead prediction . . .	145
33	Pairwise statistical results for 20-step ahead prediction . .	147
34	Average ranks for the algorithms computed on each dataset	147
35	GCN-LSTM Baseline Model Hyperparameters	152

36	AT-LSTM, AT-GRU, AT-BiLSTM, and CNN-AT-LSTM Base- line Models Hyperparameters	152
37	<i>Time-Geometric</i> Baseline Model Hyperparameters	152

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Professor Gnecco, Professor De Smedt, and Professor Parton, for their continuous guidance, encouragement, patience, and invaluable advice throughout my PhD journey. In particular, I wish to thank Professor Gnecco for always being available to answer my questions and to help correct mistakes in my papers. I am also deeply grateful to Professor De Smedt for allowing me to join the double PhD programme between KU Leuven and the IMT School for Advanced Studies Lucca, and for his support during my stay in Leuven. Finally, I would like to thank Professor Parton for his constant encouragement and the great support he provided throughout this PhD.

I am sincerely thankful to my family for their unwavering support and encouragement over the years. I would also like to thank my closest friends, who have been an incredible source of strength and positivity during this period. Their friendship, humor, and understanding have made this journey much more enjoyable. In particular, I would like to thank Federico for always giving me insightful advice, even when I sometimes ignored it; Bacio and Fra for all the laughter and jokes we shared; Jan for reminding me that my “PhD problems” were not real problems; Rosa for all the useless information he has shared with me over the years; and Anna and Antonio for providing my friends and me with a warm and welcoming place to gather.

I am also grateful to my colleagues - many of whom have become close friends - for sharing their experiences and journeys with me. In particular, I would like to thank Carlos and

Victoria for being such good friends and for hosting me in Leuven when I was homeless; Alexander for his kindness and the many funny stories he told; Brecht for making my stay in Leuven truly enjoyable with all the beers we shared; Yannis for helping me organize and supporting the most important conference in Leuven - Worbel; Yanyi for always lifting my spirits at the university; Jari for his insightful advice on my PhD projects; and Alexandre and Almer for sharing the office in Leuven and always being so friendly.

I would also like to thank my colleagues at IMT Lucca: Massimiliano and Giorgos for all the conversations and beers we had together; Fabio for being my roommate and tolerating some of my odd habits, such as waking up at six in the morning; Faskio for his genuine and kind nature; Angela and Germano for their deep and thoughtful late-night talks; Anna for our many walks along Lucca's walls; and finally, Caterina, Alessio, Niccolò, Agnese, Margherita, and Silvia for welcoming me into their group and sharing adventures that made this PhD journey much more enjoyable. There are many other colleagues, both in Lucca and in Leuven, whom I would like to thank as well. Unfortunately, due to space constraints, I cannot mention everyone by name, but I am deeply grateful to all of them for sharing their knowledge and experiences with me throughout these years.

Finally, Large Language Models (LLMs) were used solely to assist with English language editing and to improve the readability and clarity of this thesis. All scientific content, analyses, methodologies, and conclusions are the sole responsibility of the author.

*"I am glad you are here with me.
Here at the end of all things, Sam."*

Frodo Baggins,
The Lord of The Rings: The Return of the King

Vita

January 27, 1993	Born, Rome, Italy
2016	Bachelor's Degree in Economics and Finance Final mark: 93/110 University of Bologna, Bologna, Italy
2019	Master of Science in Finance Final mark: 110/110 cum laude LUISS Guido Carli, Rome, Italy
2021	Data & AI Engineering Capgemini Milan, Italy
2021	Master of Science in Data Science Final mark: 105/110 University of Rome - La Sapienza, Rome, Italy
2024	Quantitative Researcher Minerva Technologies Milan, Italy

Publications

1. M. Gregnanin, J. De Smedt, G. Gnecco, M. Parton; “Stock Price Time Series Forecasting Using Dynamic Graph Neural Networks and Attention Mechanism in Recurrent Neural Networks,” in *Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD 2023*, vol. 5, pp. 357-373, 2023.
2. M. Gregnanin, J. De Smedt, G. Gnecco, M. Parton; “Signature-Based Community Detection for Time Series,” in *Complex Networks & Their Applications XII*, vol. 1143, pp. 146-158, 2024.
3. M. Gregnanin, Y. Zhang, J. De Smedt, G. Gnecco, M. Parton; “Signature-based Portfolio Allocation: A Network Approach,” in *Applied Network Science*, vol. 9(54), 2024.
4. M. Parton, A. Fois, A. Vegliò, C. Metta, M. Gregnanin; “Predicting the Failure of Component X in the Scania Dataset with Graph Neural Networks,” in *Lecture Notes in Computer Science*, vol. 14642, pp. 252-259, 2024.
5. C. Metta, M. Gregnanin, A. Papini, S.G. Galfrè, A. Fois, F. Morandin, M. Fantozzi, M. Parton; “Achieving Predictive Precision: Leveraging LSTM and Pseudo Labeling for Volvo’s Discovery Challenge at ECML-PKDD 2024,” in *Proceedings of Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD 2024*, 2024.
6. M. Gregnanin, J. De Smedt, G. Gnecco, M. Parton; “A Generative Adversarial Graph Neural Network for Synthetic Time Series Data,” in *CEUR-WS*, vol. 3928, 2025.

Working Papers

7. M. Gregnanin, J. De Smedt, G. Gnecco, M. Parton, Li Ming; “Dynamic Hypergraph Representation Learning for Multivariate Time Series without Prior Knowledge.”
8. M. Gregnanin, J. De Smedt, G. Gnecco, M. Parton; “The Statistical Significance of Using the Signature Transform in Evaluating the Forecasting Performance of Neural Networks for Financial Time Series.”
9. M. Gregnanin, J. De Smedt, G. Gnecco, M. Parton; “Sig-Graph-Gan: A Generative Graph Neural Network for Synthetic Financial Time Series Data.”

Presentations

1. M. Gregnanin, “Stock Price Time Series Forecasting Using Dynamic Graph Neural Networks and Attention Mechanism in Recurrent Neural Networks,” at *Conference of Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD 2023*, Turin, Italy, 2023.
2. M. Gregnanin, “Signature-Based Community Detection for Time Series,” at *Conference of Complex Networks & Their Applications XII*, Menton, France, 2023.
3. M. Gregnanin, “Achieving Predictive Precision: Leveraging LSTM and Pseudo Labeling for Volvo’s Discovery Challenge at ECML-PKDD 2024,” at *Conference of Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD 2024*, Vilnius, Lithuania, 2024.
4. M. Gregnanin, “A Generative Adversarial Graph Neural Network for Synthetic Time Series Data,” at *Conference of Discovery Science*, Pisa, Italy, 2024.

Abstract

Financial time series exhibit complex statistical properties, including nonlinear dependencies, non-stationarity, and evolving relationships among assets. Traditional econometric models and purely temporal machine learning approaches often struggle to capture these characteristics as they focus mainly on sequential dynamics while overlooking the underlying geometric structure of financial data. This thesis investigates the role of geometric representations in financial time series analysis and proposes a unified framework that integrates temporal, geometric, and algebraic perspectives for modeling, forecasting, and generating financial data. The central idea is that financial time series contain latent structural information that can be captured through graph-based representations and signature-based features.

The thesis introduces a signature-based similarity measure as an alternative to correlation matrices, improving community detection and portfolio construction. It then proposes a novel Time–Geometric forecasting model that combines temporal deep learning architectures with Graph Neural Networks to capture both temporal and geometric patterns within the time series data. Finally, the framework is extended to generative modeling and higher-order dependency analysis through graph-based generative adversarial networks and hypergraph representations derived directly from multivariate time series.

Overall, this work shows that incorporating geometric structure into financial time series modeling provides a powerful framework for understanding, forecasting, and generating complex financial data, highlighting the potential of geometric deep learning in quantitative finance.

Chapter 1

Introduction

"So it begins."

Théoden,

The Lord of The Rings: The Two Towers

In the modern data-driven world, the ability to extrapolate information for modeling, forecasting, and simulating financial time series is a challenging task in computational finance. In fact, financial time series exhibits stochastic, non-linear and non-stationary behavior (Tsay, 2005; Lamberton and Lapeyre, 2011) which has led to the existence of multiple possible models for analyzing these types of time series without the existence of a universal ground truth model for the regression, classification and clustering tasks (Bishop, 2016). These characteristics make the accurate representation of financial time series both a theoretical and practical problem of great relevance, with applications ranging from algorithmic trading to portfolio optimization, systemic risk management and synthetic data generation.

Traditional statistical models - such as AutoRegressive Integrated Moving Average (ARIMA), Vector AutoRegressive (VAR) (Box et al., 2015), and Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) (Brockwell and Davis, 2002) models -, have long provided the foundation for financial time series analysis. However, these models assume that the stock prices exhibit stationary, or at the very least, weak sta-

tionary meaning that both the mean and the auto-covariance of a time series are time invariant (Tsay, 2005). On the other hand, financial mathematics models - such as Black-Scholes (Black and Scholes, 1973), Merton (Merton, 1974) models, and more generally jump processes (Tankov and Cont, 2003) -, rely on stationary and homoskedasticity assumptions while also assuming that the stock price reflects all available information of the financial market (Fama, 1970a). Even if both classical statistical and financial mathematical models have shown good results in analyzing financial time series, they often fail to capture the non-linear dependencies, regime shift, and higher-order temporal structures observed in real financial time series. As a consequence of these drawbacks, data-driven models have shown new possibilities for time series modeling. Neural network models - such as Recurrent Neural Networks (RNNs) (Goodfellow, Bengio, and Courville, 2016), Long Short-Term Memory networks (LSTM) (Hochreiter and Schmidhuber, 1997), Temporal Convolutional Networks (TCNs) (Lea et al., 2016), and Transformer networks (Vaswani et al., 2017) -, have achieved remarkable predictive capabilities by learning complex temporal patterns directly from the data. However, these models predominantly focus on capturing temporal patterns such as the autoregressive component inherent in the time series data, which can be inadequate for capturing non-stationary and particularly evolving volatility. Moreover, most deep learning models are constructed under the assumption that time series can be represented over Euclidean spaces and evolve independently of one another. This assumption overlooks the relational and geometric structure underlying financial systems, where asset prices, volatilities, or returns are interdependent and evolve on non-Euclidean manifolds that can be naturally represented as graphs (Sandhu, Georgiou, and Tannenbaum, 2016). Capturing these structures requires a new class of models capable of integrating temporal, geometric, and topological information within a unified computational framework.

The field of Geometric Deep Learning (GDL) extends classical deep learning models to non-Euclidean domains such as graphs. Among its most prominent methods, Graph Neural Networks (GNNs) (Scarselli et

al., 2008) have gained attention due to their ability to learn representations on graph-structured data. In GNNs, nodes aggregate information from their neighbors through iterative message-passing operations (W. Hamilton, Ying, and Leskovec, 2017; K. Xu, Hu, et al., 2019; Veličković, 2022), enabling the extraction of local and global structural features. In the context of financial time series, we claim that geometric patterns also play a crucial role for analyzing these types of time series. Specifically, transforming temporal data, i.e, data defined in a Euclidean space, into graph representations, i.e., data defined in a non-Euclidean space, allows the discovery of hidden geometric patterns - such as self-similarity - that complement the purely temporal dynamics leading to capture and analyze the inherent complexity of financial time series. Moreover, the adoption of a graph-based representation allows us to relax the constraint of assuming stationarity within the time series. The visibility graph algorithm (Lacasa et al., 2008), for instance, maps a univariate time series into a network where nodes represent observations and edges are created when two data points are “visible” to each other according to some geometric constraints. The graph representation facilitates the exploration of structural dynamics intrinsic to the time series, providing a means to learn relationships and dependencies among the data points. For instance, thanks to the visibility graph transformation, by analyzing the graph’s node distribution, we are able to distinguish between fractal, periodic, or random series-behaviors that are characteristic of financial dynamics. Fractals possess the capacity to encapsulate the inherent self-similarity and scaling properties that are intrinsic to financial markets (Peters, 1994; Evertsz, 1995; Mandelbrot, 2013).

However, graphs are built on pairwise relationships that can be inadequate to analyze complex systems where multiple variables may co-evolve or interact simultaneously. Hypergraphs generalize the graph-representation by allowing hyperedges that connect multiple nodes simultaneously (Bretto, 2013). This ability to connect multiple nodes allows hypergraphs to capture higher-order relationships, which often contain crucial information and characteristics about the behavior of nodes within a network. In multivariate financial systems, dependencies of-

ten extend beyond dyadic relationships - for instance, groups of assets may exhibit co-movement due to shared exposure to market sectors or macroeconomic factors. Capturing such complex dependencies requires the capacity to model multi-node interactions, which Hypergraph Neural Networks (HGNNs) (S. Bai, F. Zhang, and Torr, 2021) are equipped to handle. However, a critical challenge lies in how to derive a hypergraph representation that effectively captures these higher-order relationships within the data (Antelmi et al., 2023). Typically, a hypergraph representation relies on pre-existing knowledge about the dataset in question, as seen in applications such as citation networks (Y. Feng et al., 2019), recommendation systems (L. Li and T. Li, 2013), and traffic forecasting (Yi and Park, 2020). Alternatively, it can be derived from well-defined features or properties of data elements, such as in social networks (Gao, Y. Feng, et al., 2022). Hypergraph representations in time series contexts are often derived based on prior knowledge - such as classifications of the stock market sector (Sawhney et al., 2021) or traffic forecasting data (N. Yin, Shen, et al., 2024) -, and based on clustering the time series data (Shun Wang et al., 2024).

While GDL extends deep learning to structured domains, another complementary perspective arises from the field of path theory, which introduces the concept of the Signature Transform (Terry J Lyons, 1998; Terry J Lyons, 1998). Originally developed within the theory of rough paths, the signature provides a unique, complete, and coordinate-free representation of the evolution of a path - derived directly from the time series - through the sequence of its iterated integrals (Terry Lyons, 2014). In time series analysis, the signature transformation is able to capture both temporal patterns - such as trends, cycles, and dependencies - and geometric properties - such as loops, self-similarities, and roughness. This mathematical tool enables a shift from raw time series data to a structured feature space that encodes complex temporal dynamics in a finite-dimensional, algebraically tractable representation. Moreover, the signature can be viewed as analogous to the Moment Generating Function (MGF) in probability theory. The MGF is significant for comparing a random variable's distribution as it encodes all distribution mo-

ments into a single function, uniquely characterizing the distribution itself (Resnick, 2019), while the signature almost uniquely characterizes the path and its distribution through a hierarchy of features of increasing order. Thus, the signature serves as both a theoretical bridge between stochastic process analysis and machine learning, and a practical tool for time series representation learning (Daniel Levin, Terry Lyons, and Hao Ni, 2013).

Despite the remarkable progress achieved by recent models, a unifying framework that seamlessly integrates temporal, geometric, and algebraic representations of time series remains absent. Such a framework would not only enhance predictive accuracy but also provide interpretable and theoretically grounded insights into the structural and dynamical properties of financial systems. The overarching goal of this thesis is therefore to advance the intersection of GDL, signature analysis, and generative modeling to develop a new class of ensemble models capable of more accurately understanding, predicting, and replicating financial time series. Accordingly, the central research question addressed in this work can be formulated as follows:

How can graph-based, signature-based, and generative deep learning frameworks be integrated to effectively model, forecast, and synthesize financial time series data while capturing their temporal, geometric, and higher-order dependencies?

To address this question, we first analyze the similarity structure among financial time series. Specifically, we investigate how financial series can be grouped and how such structures can be exploited for portfolio allocation. We propose a paradigm shift from the classical correlation matrix to a similarity matrix computed on the signature of the time series (Gregarin, De Smedt, et al., 2024). The rationale for employing the signature lies in its ability to encapsulate both the temporal and geometric characteristics of the time series. We first demonstrate that the signature-based similarity matrix enhances community detection (Barabási and Pósfai, 2016) compared to communities derived from correlation matrices on the Standard & Poor’s 500 (S&P500). We then show that the community structures obtained from the signature-based similarity matrix are

highly relevant for portfolio strategy and risk management applications, as they improve diversification (Gregnanin, Y. Zhang, et al., 2024).

After establishing the usefulness of the signature transform for clustering financial time series, we shift our focus toward developing an ensemble model capable of exploiting both temporal and geometric patterns. We design a model incorporating GNNs, which effectively capture local structural information within a graph by iteratively aggregating features from neighboring nodes. This recursive aggregation allows GNNs to generalize across multiple scales, thereby modeling the intrinsic geometric properties of graphs. In our framework, the graph representation of a time series is obtained using the visibility graph algorithm, and the model is applied to forecast the future values of the S&P500 index (Gregnanin, De Smedt, et al., 2025b). We further analyze the statistical significance of performance differences between classical temporal deep learning models, - such as LSTMs, RNNs, and Gated Recurrent Unit (GRU) -, and their respective ensemble versions augmented with GNNs. Results show that the ensemble models significantly improve predictive accuracy and that these improvements are statistically meaningful when compared to their purely temporal counterparts (Marco Gregnanin et al., 2026c).

Finally, we extend our investigation toward generative modeling and higher-order network representations. First, we propose a Generative Adversarial Network (GAN) architecture combining traditional temporal deep learning components with GNN-based modules to reproduce the return distributions of multiple stock exchanges (Gregnanin, De Smedt, et al., 2025a; Marco Gregnanin et al., 2026a). To enhance the fidelity of the generated data, we incorporate a signature-based loss function that captures the higher-order distributional moments of the time series, thereby enabling the GAN to replicate the stochastic and geometric characteristics of real financial data more effectively. Subsequently, we explore how to derive hypergraph representations capable of modeling non-pairwise dependencies and latent structural patterns within multivariate time series (Marco Gregnanin et al., 2026b). These representations are further leveraged through a Hypergraph Attention Network (HAN) (S. Bai, F.

Zhang, and Torr, 2021) combined with an LSTM to forecast time series across diverse datasets.

The methodologies proposed in this dissertation are therefore motivated by the hypothesis that geometric representations can complement traditional temporal representations by encoding higher-order information about the evolution of financial time series. Rather than replacing classical temporal modeling, graph-based and signature-based representations are introduced as complementary views that expose structural information that is not readily accessible from the original sequence.

In summary, this PhD research aims to develop a unified framework that systematically integrates graph-based, signature-based, and hypergraph-based representations for modeling, forecasting, and generating financial time series. The main contributions can be summarized as follows:

- **Modeling:** Development of interpretable network and hypergraph representations of financial time series through visibility graphs and community detection, leveraging the signature transform to quantify similarities across multiple series;
- **Forecasting:** Design of hybrid architectures that integrate GNNs, LSTMs, and attention mechanisms to capture both temporal and geometric dependencies in univariate and multivariate settings;
- **Generation:** Introduction of generative models combining the signature transform and hybrid deep architectures to produce realistic synthetic financial time series with controllable statistical and fractal properties;
- **Evaluation:** Rigorous statistical significance testing and financial performance assessment of the proposed models;
- **Application:** Demonstration of the practical relevance of the proposed framework in stock price forecasting, portfolio optimization, risk management, and market simulation.

The remainder of this thesis is organized as follows: Chapter 2 reviews the existing literature relevant to this research; Chapter 3 describes the

main methodological tools employed; Chapter 4 investigates the role of the signature transform in comparing multiple time series and its applications to community detection and asset allocation; Chapter 5 presents the proposed hybrid forecasting models, their empirical evaluation, and their statistical comparison with traditional temporal models; Chapter 6 focuses on the generation of synthetic financial time series and the derivation of hypergraph representations from multivariate data; finally, Chapter 7 concludes the thesis and outlines potential directions for future research.

Chapter 2

Related Works

*"I don't know half of you half as well as I should like,
and I like less than half of you half as well as you deserve."*

Bilbo Baggins,
The Lord of The Rings: The Fellowship of the Ring

Research on financial time series modeling has progressively evolved from traditional econometric frameworks to advanced machine learning approaches and, more recently, to geometric and topological deep learning paradigms. This section reviews the relevant literature across five key dimensions: classical statistical and stochastic models; deep learning architectures for temporal prediction; graph and hypergraph networks for time series representation; network-theoretic approaches for community detection and portfolio allocation; and signature-based as well as generative methodologies for the representation and synthesis of financial data. Collectively, these strands form the theoretical and methodological foundation that motivates the unified framework proposed in this thesis.

2.1 Financial Time Series Forecasting from Statistical Models to Machine Learning

The modeling of financial time series has traditionally relied on statistical and stochastic process frameworks that assume stationarity and linear dependence. Statistical models aim to discern and encapsulate distinct components within a time series, namely Trend, Seasonality, and Noise (Box et al., 2015). The AutoRegressive Moving Average (ARMA) model (Brockwell and Davis, 1991) serves as a foundational statistical model for univariate time series analysis, combining autoregressive components with moving average elements. In a study by Rounaghi and Zadeh (2016), the ARMA model was employed for predicting stock returns in both the Standard and Poor's 500 (S&P500) and the London Stock Exchange. However, limitations arise as the ARMA model assumes stationarity within the time series, presupposing constancy in mean and variance over time. The AutoRegressive Integrated Moving Average (ARIMA) model (J. D. Hamilton, 2020) extends the ARMA model by not explicitly assuming time series stationarity. Instead, the integrated component introduces differencing in the time series, achieving stationarity therein. Ariyo, Adewumi, and Ayo (2014) applied the ARIMA model to predict the New York Stock Exchange (NYSE) and the Nigeria Stock Exchange (NSE). Nevertheless, the ARIMA model reveals its limitations when faced with non-constant variance over time because ARIMA captures linear dependencies within a time series. The Vector AutoRegressive (VAR) model extends this concept to capture linear interdependencies among multiple time series. In Leung, Daouk, and A.-S. Chen (2000), the VAR model was used to predict the future value of six different stock exchanges. To address the challenge of evolving volatility, the AutoRegressive Conditional Heteroskedasticity (ARCH) model was introduced (Engle, 1982; Bollerslev, Engle, and Nelson, 1994). In a study by Engle (1982), the ARCH model was employed to estimate the inflation mean and variance in the United Kingdom from 1958 to 1977, while in Bollerslev, Chou, and Kroner (1992) the ARCH model was used on financial data for studying the persistence of the variance. However,

the ARCH model depends only on using a few past error terms to model variance that can lead to less flexibility to capture longer-term volatility patterns. Mitigating this limitation, the Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) model was proposed (Bollerslev, 1986; Francq and Zakoian, 2019). Lee, Nguyen, and Sy (2017) conducted a comparative analysis between the ARIMA and GARCH models for predicting volatility in various stock exchanges.

On the other hand, classical mathematical finance was based on the Geometric Brownian Motion (GBM) proposed by Bachelier in his PhD thesis in 1900 (Lamberton and Lapeyre, 2011). The GBM and its derivation was typically used to model the price fluctuations of an asset. In Black and Scholes (1973) and Merton (1974) the GBM was used to derive the theoretical price of stock options; while in Abidin and Jaffar (2014) and Farida Agustini, Affianti, and Putri (2018) the GBM was used to predict the future value of different asset classes achieving satisfactory results for short-term forecasting. However, the GBM modeled asset dynamics in continuous time, which can be limiting in capturing market discontinuities and volatility clustering. For these reasons, jump processes, and more in general, Levy processes (Tankov and Cont, 2003) were proposed. In Mba, Mwambi, and Pindza (2022), the Ornstein–Uhlenbeck Levy Process was used to predict the Bitcoin price. While foundational, these models impose strong parametric assumptions that limit their ability to represent the non-linear and fractal properties observed in real markets (Mandelbrot, 2013).

In contrast to statistical and classical financial mathematics models, deep learning models offer the advantage of not requiring stationarity assumptions in time series data, and a direct underlying mathematical model that describes the time series behavior. Furthermore, deep learning models have demonstrated notable efficacy in capturing intricate non-linear patterns within such data. Among the class of neural network models tailored for time series analysis, Recurrent Neural Networks (RNNs) (Goodfellow, Bengio, and Courville, 2016) represent a fundamental model designed for sequential data processing. Guo et al. (2016) applied RNNs for time series forecasting within the domain of

anomaly detection. However, RNNs face challenges associated with the vanishing gradient problem, limiting their ability to capture long-term relationships within a time series. To address this limitation, the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) model was introduced, along with its simplified counterpart, the Gated Recurrent Unit (GRU) (J. Chung et al., 2014). Ghosh et al. (2019) utilized LSTMs for predicting Indian Stock Exchanges, while C. Chen, L. Xue, and Xing (2023) compared the GRU model with other machine learning models for stock price prediction across various stock exchanges. Additionally, Roondiwala, Patel, and Varma (2017) employed LSTMs to predict the NIFTY50 index. L. Zhang, Aggarwal, and Qi (2017) enhanced the LSTM model’s performance by integrating the Discrete Fourier Transform into the memory cell. Furthermore, an attention mechanism was incorporated to augment the predictive capability of the LSTM model for various stock exchanges in Lin et al. (2022). Selvin et al. (2017) conducted a comparative analysis of the forecasting accuracy of LSTM, RNNs, and Convolutional Neural Networks (CNNs) on the Indian Stock Exchange. Finally, ensemble methods were proposed for predicting the Shanghai Composite Index in Ding et al. (2020) and Lu et al. (2021). Applications to financial forecasting demonstrated that these models outperform traditional statistical and financial mathematics models but often overfit or fail to generalize under regime shifts due to the intrinsic stochasticity of markets (Sezer, Gudelek, and Ozbayoglu, 2020). Furthermore, they typically process temporal information in Euclidean spaces and ignore the relational geometry among assets, thereby overlooking important structural information.

2.2 Graph Neural Networks in Time Series

The extension of deep learning methodologies to graph-structured data has given rise to the field of Geometric Deep Learning (GDL), which generalizes classical convolutional and recurrent operations from Euclidean domains to non-Euclidean spaces, such as graphs, and manifolds. Unlike traditional deep learning architectures, which assume data to be repre-

sented on regular grids, GDL provides a framework for learning from data characterized by irregular structures and relational dependencies. Within this paradigm, the most prominent and widely adopted class of models is that of Graph Neural Networks (GNNs) (Scarselli et al., 2008). GNNs are specifically designed to capture geometric and topological patterns by exploiting the relational structure encoded in graphs. They operate by iteratively aggregating and transforming information from a node’s local neighborhood through a message-passing or neighborhood aggregation mechanism, thereby enabling the learning of expressive node- and graph-level representations. This iterative process allows GNNs to propagate information across the graph, capturing both local and global structural properties in a data-driven manner. As a result, GNNs have become a foundational tool for modeling data that naturally reside in non-Euclidean spaces. Due to their flexibility and expressive power, GNNs have been successfully applied across a wide range of domains, including traffic forecasting, social network analysis, graph classification, recommendation systems, and the analysis of biological and chemical data (Z. Wu, Pan, F. Chen, et al., 2020). Depending on whether the underlying graph structure remains fixed or evolves over time, GNN-based models can be broadly categorized into static and dynamic architectures (Z. Wu, Pan, F. Chen, et al., 2020). Static GNNs assume a fixed topology throughout the learning process, whereas dynamic GNNs are designed to accommodate changes in node attributes, edge weights, or even the graph structure itself, thereby making them particularly suitable for temporal and sequential data.

The fundamental building block underlying most GNN architectures is the Graph Convolutional Network (GCN) (Kipf and Welling, 2017), originally introduced to address semi-supervised node classification tasks on citation and knowledge graphs. GCNs extend the notion of convolution to graphs by defining localized spectral or spatial filtering operations, enabling the aggregation of information from neighboring nodes while preserving the graph structure. Several influential extensions of the GCN framework have subsequently been proposed to improve scalability, expressiveness, and learning efficiency. Among the most notable

implementations are GraphSAGE (SAmple and aggreGatE) (W. Hamilton, Ying, and Leskovec, 2017) and the Graph Attention Network (GAT) (Veličković et al., 2018). GraphSAGE enhances the original GCN formulation by introducing an inductive learning framework that efficiently samples and aggregates features from a node’s neighborhood, allowing the model to generalize to previously unseen nodes and graphs. In contrast, GAT incorporates an attention mechanism into the aggregation process, enabling the model to assign different levels of importance to neighboring nodes. This adaptive weighting mechanism allows GATs to focus on the most relevant relationships in a node’s local neighborhood, thereby improving representational capacity and interpretability.

In the context of time series analysis, GNNs have demonstrated particular effectiveness in spatio-temporal modeling, where observations are not only temporally correlated but also interconnected through an underlying graph structure. Prominent applications include traffic flow prediction, social network dynamics, and epidemic modeling, where both spatial dependencies and temporal evolution play a crucial role. However, time series forecasting tasks pose additional challenges, as models must account for both temporal dependencies and the evolving nature of graph structures. To address these challenges, Dynamic Graph Neural Networks (DGNNs) have been introduced and extensively studied in the context of multivariate time series analysis. In particular, to capture the time-varying nature of graph topology, DGNNs typically adopt a snapshot-based representation, whereby the graph structure is updated at each time step (Skarding, Gabrys, and Musial, 2021). Several notable DGNN architectures have been proposed in the literature. The Spatio-Temporal Graph Convolutional Network (STGCN) (Yu, H. Yin, and Z. Zhu, 2018) integrates temporal convolutional layers with graph convolutions to model traffic dynamics. The Spectral Temporal GNN (StemGNN) (Cao et al., 2020) leverages spectral graph theory to capture both spatial and temporal dependencies and has been evaluated on real-world datasets, including traffic forecasting and COVID-19 time series. Similarly, the Temporal Graph Convolutional Network (T-GCN) (L. Zhao et al., 2019) combines graph convolutions with recurrent archi-

tectures to model temporal evolution on graphs. Other approaches include Graph Wavelet Neural Networks (GWNNs) (Z. Wu, Pan, Long, et al., 2019), which utilize graph wavelets to perform multi-scale analysis, and Graph Recurrent Neural Networks (GRNNs) (L. Ruiz, Gama, and Ribeiro, 2020), which incorporate gating mechanisms to model dynamic processes such as earthquake epicenter localization, traffic forecasting, and epidemic spread. In the financial domain, GNN-based models have increasingly been adopted to capture complex interdependencies among assets. The Multi-Modality Graph Neural Network (MAGNN) (D. Cheng et al., 2022) integrates heterogeneous data sources to analyze two major Chinese stock exchanges, demonstrating the effectiveness of graph-based representations for market modeling. Similarly, Graph WaveNet with Self-Attention (G. Zhao, M. Xue, and L. Cheng, 2023) extends the GWNN framework by incorporating attention mechanisms to predict crude oil prices, thereby enhancing the model’s ability to capture long-range dependencies. The Temporal and Heterogeneous GNN (THGNN) (Xiang et al., 2022) focuses on modeling heterogeneous financial relationships and has been applied to predict market behaviors in both the United States and Chinese stock markets.

Despite their success, the majority of existing GNN-based financial models are primarily designed for multivariate time series and rely on predefined graph structures, typically derived from correlation matrices, attention-based similarity measures, or geographical and sectoral information. As a consequence, these approaches often assume the availability of reliable prior knowledge about asset relationships and may fail to capture intrinsic geometric patterns present within individual time series. In contrast, alternative approaches have emerged that derive graph representations directly from univariate time series data, without relying on external relational information. In particular, Lazcano, Herrera, and Monge (2023) proposed constructing graph representations of crude oil prices using the visibility graph method (Lacasa et al., 2008), which transforms a time series into a graph based on geometric visibility criteria. This approach enables the extraction of structural patterns inherent in the temporal evolution of the series and facilitates the application

of GNNs to univariate forecasting tasks. Such methodologies highlight the potential of graph-based representations to uncover latent geometric structures in financial time series, thereby motivating further investigation into the integration of temporal and geometric learning paradigms.

2.3 Signature-Based and Hypergraph-Based Representation Learning for Time Series

The increasing complexity of real-world time series data, particularly in domains such as finance, has motivated the development of representation learning techniques that move beyond purely temporal or pairwise modeling assumptions. Two complementary paradigms have emerged in this context: signature-based representations, originating from rough path theory, and hypergraph-based learning, which generalizes graph models to capture higher-order relationships.

Introduced by Terry J Lyons (1998), the signature of a path provides a universal, non-parametric, and coordinate-free representation of sequential data. Unlike traditional feature extraction techniques, which rely on handcrafted statistics or fixed temporal windows, the signature transform encodes the entire evolution of a time series into a structured algebraic object that uniquely characterizes the underlying path up to reparameterization (Terry Lyons, 2014). This property allows the signature to capture both temporal ordering and geometric structure in a mathematical manner. From a machine learning perspective, the signature transform establishes a powerful connection between stochastic process theory and representation learning. Early applications of signatures were primarily situated within stochastic analysis and the theory of controlled differential equations (Terry Lyons, 2014). More recently, however, a growing body of work has demonstrated their effectiveness as feature representations for data-driven learning tasks, including classification, regression, and sequence modeling across a wide range of application domains. In particular, signature-based representations have been successfully applied to handwriting recognition (Graham, 2013), healthcare and clinical time series analysis (A. B. Kormilitzin et al., 2016; A. Kormil-

itzin et al., 2017; P. J. Moore, T. J. Lyons, and Gallacher, 2019; Rehman et al., 2020), motion capture and human activity modeling (W. Yang et al., 2022), and more general data analysis tasks (Chevyrev, Nanda, and Oberhauser, 2020). These applications highlight the ability of the signature transform to capture long-range dependencies and higher-order temporal interactions in sequential data. In the context of time series analysis, the signature transform has attracted increasing attention due to its generality as a representation for arbitrary sequential data. For instance, Morrill et al. (2021) employed generalized signature features for time series classification, demonstrating competitive performance across multiple benchmark datasets. Within the financial domain, the signature transform has been applied to problems that require non-parametric modeling of complex temporal dynamics. In particular, Terry Lyons, Nejad, and Arribas (2020) utilized signatures to develop a non-parametric framework for the pricing and hedging of exotic derivatives, while Kalsi, Terry Lyons, and Arribas (2020) applied signature-based representations to approximate solutions to optimal execution problems. These studies illustrate the suitability of the signature transform for modeling high-dimensional, non-linear, and non-stationary financial time series. More recently, the signature transform has been combined with deep learning architectures, giving rise to hybrid models that integrate algebraic feature extraction with neural network-based learning. In Kidger, Morrill, et al. (2020), the signature transform was shown to act as a universal approximation mechanism for Neural Controlled Differential Equation (Neural CDE) models, establishing a theoretical link between signature features and continuous-time deep learning architectures. Subsequently, Bonnier et al. (2019) investigated the integration of the signature transform with a variety of deep learning architectures, proposing a Deep Signature framework in which signature features can be incorporated as a differentiable layer at arbitrary depths within a neural network. This approach highlights the flexibility of the signature transform as a modular representation-learning component that can be seamlessly combined with existing deep learning models. More recently, A. Tong et al. (2023) combined the signature transform with Transformer architectures, intro-

ducing a hybrid model for hedging strategies applied to the S&P500. Similarly, Moreno-Pino et al. (2024) proposed a Rough Transformer architecture, in which path signatures are coupled with attention mechanisms to jointly capture local and global dependencies in sequential data. The proposed framework was evaluated on a range of time series tasks, demonstrating the effectiveness of combining signature-based representations with attention-based deep learning models.

While signature transforms are highly effective in representing individual time series, many real-world applications - especially in finance - require modeling collections of interacting time series. Financial markets, for instance, consist of groups of assets whose dynamics are interdependent and often governed by collective behaviors that cannot be reduced to pairwise interactions. Capturing such dependencies necessitates a relational representation that extends beyond classical graphs. Hypergraph representation learning provides a natural generalization of graph-based models by allowing hyperedges that connect more than two nodes simultaneously.

Existing hypergraph representation learning methods aim to construct hypergraphs that are capable of capturing higher-order relationships within data, extending beyond the pairwise interactions modeled by standard graphs. Early work in this area focused on domains where such high-order interactions naturally arise. In computer vision, Gao, M. Wang, et al. (2012) derived hypergraph structures by clustering multiple views of a 3-D object, while S. Huang et al. (2015) constructed hypergraphs based on shared attributes among objects. Similarly, Jin et al. (2019) proposed deriving hyperedges through ridge regression in order to denoise image representations, and Y. Feng et al. (2019) employed a clique-based strategy that connects K -Nearest Neighbors (KNN) to a selected node for object classification tasks. Subsequent research has focused on generalizing graph neural network frameworks to hypergraph settings. The Uni-GNN model proposed by Jing Huang and J. Yang (2021) introduced a unified formulation that bridges GNNs and Hypergraph Neural Networks (HGNNs) (S. Bai, F. Zhang, and Torr, 2021), enabling hypergraphs to be constructed directly from data attributes. In a related manner,

Gao, Y. Feng, et al. (2022) proposed a hybrid hypergraph construction strategy that adapts to the availability of an underlying graph structure: when such structure exists, a clique-based method is employed, whereas in its absence, hyperedges are derived using node attributes and a KNN algorithm. Hypergraph representation learning has also been applied in other domains. In social network analysis, X. Sun et al. (2023) constructed hypergraphs based on interactions among individuals, while Ma, W. Zhao, and Z. Yang (2024) introduced a directed hypergraph formulation to address link classification across multiple datasets. More recently, S. Li et al. (2025) employed a KNN-based hypergraph construction on tabular data to support anomaly detection tasks, further demonstrating the versatility of hypergraph-based representations. In the context of time series analysis, hypergraph representations are often constructed using prior domain knowledge. For instance, Yi and Park (2020) derived hypergraph structures from network sensor data and combined Hypergraph Convolutional Networks (HCNs) with Recurrent Neural Networks (RNNs) to model temporal dynamics. A similar reliance on prior structural information can be observed in traffic forecasting applications. In Luo, Peng, and Liang (2022), a hypergraph was constructed from an underlying traffic network and analyzed using directed Hypergraph Attention Networks (HANs), while J. Wang et al. (2022) derived hypergraph representations from bus and metro transportation networks to predict metro flow using HCNs followed by temporal modeling layers. In financial applications, Sawhney et al. (2021) constructed hypergraph representations over baskets of stocks based on sector classifications and corporate relationships, addressing a stock return ranking problem using HANs. While these approaches demonstrate the effectiveness of hypergraph-based models, they typically rely on fixed hypergraph structures that remain unchanged throughout the analysis period, limiting their ability to capture evolving dependencies in time-varying systems.

To model systems in which relationships evolve over time, dynamic hypergraph representation learning has been proposed, allowing both nodes and hyperedges to change as functions of time. An early dynamic hy-

pergraph learning framework was introduced by Z. Zhang et al. (2018), focusing on learning label projections and data attributes in the context of 3-D shape and gesture recognition. In Jiang et al. (2019), dynamic hypergraph representations for citation networks and social data were constructed using a combination of k -means clustering and KNN, and subsequently analyzed using Hypergraph Convolutional Networks (HCNs) for classification tasks. Similarly, Kang et al. (2022) adopted a comparable approach to generate and analyze dynamic hypergraphs across multiple datasets, including citation networks and traffic flow data. More recent work has extended dynamic hypergraph learning to time series settings. In N. Yin, F. Feng, et al. (2022) and N. Yin, Shen, et al. (2024), dynamic hypergraph representations were derived from the Stocktwits and Tiingo datasets for stock price prediction, as well as from the NYC taxi dataset for taxi demand forecasting, using HGNN-based architectures. In Xia et al. (2024), dynamic hypergraph representations of stock returns were constructed based on company sector information, corporate relationships, and Dynamic Time Warping (DTW) similarity measures, enabling stock ranking via HANs. Similarly, Shun Wang et al. (2024) proposed constructing dynamic hypergraph representations across six distinct datasets - including wind, temperature, and metro data - by combining KNN-based hypergraph construction with graph structures derived from attribute knowledge. Although these dynamic hypergraph-based models demonstrate improved predictive performance over static counterparts, they generally do not explicitly account for how time series may cluster based on their historical observations and intrinsic similarities. As a result, the induced hypergraph structures may fail to fully capture latent temporal-geometric relationships among time series, highlighting an important limitation and motivating further research into data-driven and geometry-aware hypergraph construction methods.

2.4 Generative Models for Synthetic Financial Time Series

The generation of synthetic financial time series is a problem of significant practical and theoretical importance, primarily due to the limited availability of real financial data. While low-frequency data, such as daily prices and returns, are generally accessible, high-frequency financial data are often proprietary and costly. Relying exclusively on publicly available low-frequency data can therefore lead to biased models and overfitting, particularly in data-intensive machine learning applications (Assefa et al., 2020). As a result, the development of reliable methods for generating realistic synthetic financial time series has become an active area of research.

Broadly, existing approaches for synthetic financial time series generation can be divided into model-based and data-driven methodologies. Model-based approaches rely on statistical and mathematical formulations of the data-generating process. Classical statistical models - ARIMA, ARCH, and GARCH - differ primarily in their treatment of volatility, with ARIMA assuming constant variance and ARCH/GARCH allowing for time-varying volatility. ARCH- and GARCH-type models have been widely applied to model and forecast volatility in financial markets such as the Nasdaq and Dow Jones indices (Engle, Focardi, and Fabozzi, 2012). In continuous time, mathematical finance models predominantly rely on GBM. The Black-Scholes-Merton framework (Black and Scholes, 1973) models asset prices as GBM processes and has been extensively used for pricing European options. Subsequent extensions introduced jumps to capture fat-tailed return distributions (Merton, 1976; Kou, 2002), as well as stochastic volatility dynamics (Heston, 1993). While these models are mathematically tractable and interpretable, they impose strong assumptions - such as stationarity, normality, or specific volatility structures - that often fail to reflect the complex, non-stationary behavior of real financial markets. To alleviate some of these restrictions, resampling-based methods such as Monte Carlo simulations (H. Wang, 2012) and bootstrapping techniques (E. Ruiz and Pascual, 2002) have been employed

to generate synthetic price paths and return distributions. Although these approaches preserve certain empirical properties of the observed data, they typically inherit assumptions from the underlying models and are limited in their ability to generate genuinely novel trajectories or capture long-range dependencies.

The limitations of model-based and resampling approaches have motivated the adoption of data-driven generative models, particularly within the deep learning paradigm. Among these, Generative Adversarial Networks (GANs) (Goodfellow, Pouget-Abadie, et al., 2020) have emerged as a powerful framework for learning complex data distributions. GANs have been adapted to sequential data through architectures such as SeqGAN (Lantao et al., 2017), which targets discrete sequential data, and C-RNN-GAN (Mogren, 2016), which employs recurrent neural networks to generate musical sequences. To explicitly preserve temporal dynamics in continuous-valued time series, TimeGAN (Yoon, Jarrett, and Schaar, 2019) introduced a hybrid adversarial-supervised framework that enforces consistency in latent temporal representations. In the financial domain, Quant GANs (Wiese et al., 2020) leverage Temporal Convolutional Networks (TCNs) to capture long-term dependencies in asset price and return series. Alternative generative approaches include Restricted Boltzmann Machines (RBMs), which have been used to generate synthetic market data via stochastic binary latent representations (Konratyev and Schwarz, 2019). Several improvements to GAN training stability and distributional fidelity have also been proposed. In particular, the Wasserstein loss was introduced to measure distances between probability distributions more robustly (De Meer Pardo, 2019). Building on this idea, Hao Ni et al. (2020) proposed integrating GANs with the signature transform by introducing the Signature Wasserstein-1 (Sig- W_1) metric as a loss function, thereby aligning real and generated time series in terms of higher-order temporal structure. This framework was subsequently extended to multi-stock price generation in (De Meer Pardo, Schwendner, and Wunsch, 2022). More recently, DAT-CGAN (H. Sun et al., 2023) was proposed to generate synthetic financial time series tailored to multi-period asset allocation problems.

Despite these advances, most existing generative models for financial time series focus predominantly on the autoregressive nature of sequential data. Moreover, such approaches often operate in Euclidean spaces and primarily aim to reproduce marginal distributions or short-term temporal dependencies. Consequently, they tend to neglect the geometric structure and higher-order temporal patterns that characterize financial time series, such as fractal behavior, volatility clustering, and regime-dependent dynamics.

2.5 Summary and Research Gap

The literature reviewed in this chapter demonstrates significant progress in the modeling, representation, and generation of financial time series. Classical statistical and econometric models, and stochastic differential equation-based formulations, provide mathematically tractable descriptions of market dynamics but rely on restrictive assumptions that limit their effectiveness in non-stationary and highly nonlinear financial environments. Advances in deep learning have substantially improved time series forecasting, with recurrent, convolutional, and attention-based architectures capturing complex temporal dependencies. More recently, geometric deep learning has extended these models to non-Euclidean domains, enabling the explicit modeling of relational and structural information through static and dynamic graph neural networks. These approaches have proven particularly effective in multivariate financial settings, where inter-asset dependencies play a critical role. Parallel to these developments, the signature transform has emerged as a powerful representation-learning framework for sequential data, providing a universal and non-parametric encoding of temporal and geometric structure. Signature-based methods overcome key limitations of correlation-based measures and have shown strong empirical performance in financial networks, community detection, and portfolio optimization. To further capture collective behaviors, hypergraph representation learning generalizes graph-based models by modeling higher-order interactions and their temporal evolution, offering richer representations of multi-

variate time series systems. In the context of synthetic financial time series generation, research has evolved from parametric stochastic models and resampling techniques to deep generative frameworks, particularly GAN-based architectures. While these methods offer increased flexibility, many focus primarily on autoregressive dynamics and marginal distributions, often neglecting geometric structure and evolving relational dependencies. Recent structure-aware approaches incorporating graph-based representations and signature-based loss functions represent an important step toward more realistic data generation.

Despite these advances, several key gaps remain unaddressed in the existing literature. First, current approaches to financial time series analysis tend to treat forecasting, representation learning, and synthetic data generation as largely independent problems. While individual methods incorporate temporal, geometric, or relational information, there is a lack of a unified framework that systematically integrates these perspectives within a single modeling paradigm. Second, although graph-based and hypergraph-based models capture relational structure, the construction of these structures often relies on prior domain knowledge, static correlations, or predefined similarity measures. Conversely, while signature-based representations effectively capture temporal and geometric properties of individual time series, their integration with dynamic relational models remains limited. Third, existing generative models for financial time series predominantly focus on reproducing autoregressive behavior and marginal distributions. Only a limited number of studies incorporate geometric or higher-order relational constraints, and even fewer address the generation of multivariate time series with time-varying dependency structures. As a result, synthetic data often fail to preserve collective dynamics observed in real markets.

Chapter 3

Methodology

*"Anyway, You Need People of Intelligence
on This Sort of Mission, Quest, Thing."*

Pippin,

The Lord of The Rings: The Fellowship of the Ring

In this chapter, we present the theoretical and methodological foundations underlying the models employed throughout this thesis. We begin by introducing the fundamental concepts of graph theory, with particular emphasis on the visibility graph algorithm, which plays a central role in the transformation of time series data into graph-based representations. We then review the principal deep learning frameworks relevant to this work. Specifically, we examine the Graph Neural Network (GNN) paradigm, discussing its core principles and the main model formulations derived from it. Finally, we provide a comprehensive mathematical definition and detailed explanation of the Signature Transform, highlighting its theoretical properties and its role as a representation-learning tool for sequential data.

3.1 Graph Theory and Visibility Graph

Let $G = (V, E)$ denote a graph, where $V = \{v_1, \dots, v_N\}$ is the set of N nodes, and $E \subseteq V \times V$ is the set of edges representing relationships

between pairs of nodes. Unless stated otherwise, G is assumed to be a simple, unweighted, and undirected graph. The structure of G can be encoded by its adjacency matrix $A \in \mathbb{R}^{N \times N}$, whose entries indicate whether two nodes are connected by an edge. Formally, the entries of A are defined as:

$$A(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

for all $u, v \in V$. Since the graph is undirected, the adjacency matrix is symmetric, i.e., $A(u, v) = A(v, u)$. The adjacency matrix provides a compact algebraic representation of the graph and constitutes the foundation for many graph-theoretic analyses and graph-based learning algorithms. A fundamental local property of a node $v \in V$ is its degree, denoted by K_v , which measures the number of edges incident to v . Mathematically, the degree of node v can be computed directly from the adjacency matrix as:

$$k_v = \sum_{u \in V} A(u, v)$$

The node degree reflects the immediate connectivity of a node and plays a central role in characterizing both local and global structural properties of the network.

An important global characteristic of a graph is the degree distribution, which provides insight into the overall topology of the network. The degree distribution, denoted as $\mathbb{P}(k)$, is defined as the probability that a randomly selected node v has degree k , namely:

$$\mathbb{P}(k) = \frac{1}{|V|} |\{v \in V : k_v = k\}|$$

The degree distribution is commonly used to distinguish between different classes of networks, such as random graphs, small-world networks, and scale-free networks. In many real-world systems, including financial and social networks, the degree distribution exhibits heavy-tailed behavior, indicating the presence of highly connected nodes, often referred to as hubs (Barabási and Pósfai, 2016).

The network structure can be analyzed through community detection,

which aims to identify groups of nodes that are more densely connected to each other than to the rest of the network. A widely used measure to quantify the structure of the community is modularity. Given a partition of the node set into communities, modularity $Q(\eta)$ is defined as:

$$Q(\eta) = \frac{1}{2|E|} \sum_{u,v \in V} \left(A(u,v) - \frac{k_u k_v}{2|E|} \right) \delta(\eta_u, \eta_v) \quad \in [-0.5, 1] \quad (3.2)$$

where k_u and k_v denote the degrees of nodes u and v , respectively, η_u denotes the community assignment of node u , and $\delta(\cdot, \cdot)$ is the Kronecker delta function, which is equal to 1 when $\eta_u = \eta_v$ and 0 otherwise. Modularity measures the extent to which the observed density of edges within communities exceeds that expected under a null model that preserves the degree distribution. High modularity values indicate a strong and well-defined community structure.

The visibility graph algorithm (Lacasa et al., 2008) provides a method for transforming a univariate time series into a graph representation by establishing connections between data points that satisfy a geometric visibility criterion. Intuitively, the algorithm represents the values of a time series as vertical bars and connects two bars if there exists an unobstructed line of sight between their tops. Under this construction, each observation in the time series corresponds to a node in the graph, while edges encode visibility relationships between pairs of observations. Formally, let $S = \{s_1, s_2, \dots, s_T\}$ denote a univariate time series observed at discrete time stamps $\mathcal{T} = \{t_1, t_2, \dots, t_T\}$. Two data points (s_i, t_i) and (s_j, t_j) are connected by an edge in the visibility graph if and only if every intermediate point (s_k, t_k) satisfies the following visibility condition:

$$s_k < s_i + (s_j - s_i) \frac{t_k - t_i}{t_j - t_i} \quad \forall t_i < t_k < t_j. \quad (3.3)$$

This condition ensures that the straight line connecting (s_i, t_i) and (s_j, t_j) lies strictly above all intermediate data points, thereby guaranteeing mutual visibility. In the context of time series analysis, the visibility graph transformation captures non-linear dependencies and structural patterns

that are not readily observable in the original Euclidean representation (Lacasa et al., 2008). Moreover, the resulting graph is invariant under affine transformations of the time series and remains connected under broad conditions (Stephen, Gu, and H. Yang, 2015), making the method robust to scaling and translation of the data. The visibility graph algorithm can be used to construct either undirected or directed graphs. In the undirected case, the visibility criterion in (3.3) is evaluated for all pairs of observations (s_i, t_i) . In contrast, directed visibility graphs enforce a temporal ordering by applying the visibility condition only in a forward-time manner, that is, by connecting each point (s_i, t_i) to subsequent observations (s_j, t_j) , with $t_j > t_i$. This “from left to right” formulation preserves the causal structure inherent in time series data.

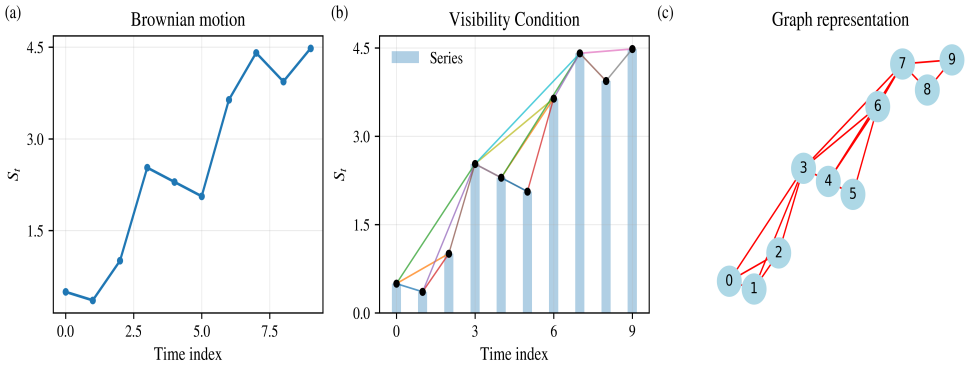


Figure 1: Visibility graph algorithm applied to a standard Geometric Brownian Motion with 10 observations.

Figure 1 demonstrates the application of the visibility graph algorithm to a univariate time series, computed using the “Time series to visibility graphs” (ts2vg) Python package (Bergillos, 2020). The first plot illustrates a Geometric Brownian motion trajectory, the middle plot depicts the visibility condition and link formation, and the final plot presents the corresponding undirected graph.

A notable property of the visibility graph transformation is its ability to

preserve key structural characteristics of the original time series. Specifically, periodic time series are mapped to regular graphs, in which each node has the same degree; random time series give rise to random graphs; and fractal time series are represented as small-world networks (Lacasa et al., 2008). As a result, the visibility graph provides an effective representation for non-stationary time series, preserving temporal information while capturing intrinsic patterns such as peaks, troughs, and long-term trends. This makes the visibility graph a valuable tool for bridging time series analysis and graph-based learning methods.

3.2 Graph Neural Networks

GNNs constitute a broad class of deep learning models designed to learn expressive representations from graph-structured data. By operating directly on non-Euclidean domains, GNNs enable the extraction of node-, edge-, or graph-level embeddings that capture both local neighborhood information and global structural properties. The predominant paradigm underlying GNNs is message passing, whereby nodes iteratively exchange information with their neighbors to update their latent representations (Gilmer et al., 2017; K. Xu, C. Li, et al., 2018).

Following the general formalism presented in You, Du, and Leskovec (2022), the computation performed at the l -th layer of a GNN can be expressed as

$$\begin{aligned}\mathbf{m}_{u \rightarrow v}^{(l)} &= \text{MSG}^{(l)} \left(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)} \right), \\ \mathbf{h}_v^{(l)} &= \text{AGG}^{(l)} \left(\mathbf{m}_{u \rightarrow v}^{(l)} | u \in \mathcal{N}(v), \mathbf{h}_v^{(l-1)} \right).\end{aligned}\quad (3.4)$$

where $\mathbf{h}_v^{(l)}$ denotes the embedding of node v at layer l , with the initialization $\mathbf{h}_v^{(0)} = \mathbf{x}_v$, the input feature vector associated with node v . The function $\text{MSG}^{(l)}(\cdot)$ defines the message function, which computes messages transmitted from neighboring nodes, while $\mathcal{N}(v)$ denotes the set of neighbors of node v . The function $\text{AGG}^{(l)}(\cdot)$ represents the aggregation function, responsible for combining incoming messages with the previous representation of node v . The message function governs how in-

formation is exchanged between nodes and their neighbors and is often implemented as a learnable linear or affine transformation of node embeddings. The aggregation function must be permutation invariant with respect to the ordering of neighboring nodes to ensure consistent representations. Common aggregation operators include summation, mean, and maximum functions. By stacking multiple message-passing layers, a GNN enables information to propagate across the graph, such that the representation $\mathbf{h}_v^{(l)}$ encodes information from nodes that are at most l hops away from v . This hierarchical information diffusion allows GNNs to capture increasingly global structural patterns as depth increases. The message-passing operations defined in (3.4) describe how Graph Neural Networks propagate and update information at the node level. For several GNN architectures, these operations can be equivalently expressed in a matrix formulation, which is advantageous for efficient implementation and facilitates theoretical analysis. In this setting, the node embeddings at layer l are stacked into a matrix:

$$H^{(l)} = \begin{bmatrix} \left(\mathbf{h}_1^{(l)}\right)^T \\ \left(\mathbf{h}_2^{(l)}\right)^T \\ \vdots \\ \left(\mathbf{h}_n^{(l)}\right)^T \end{bmatrix} \in \mathbb{R}^{n \times d_l}$$

where n denotes the number of nodes in the graph and d_l is the dimensionality of the hidden representations at layer l . Each row of $H^{(l)}$ corresponds to the embedding of a single node, thereby providing a compact representation of all node features at that layer.

A fundamental and widely used GNN architecture is the Graph Convolutional Network (GCN) (Kipf and Welling, 2017). GCNs generalize the notion of convolution from Euclidean domains to graph-structured data by defining localized spectral or spatial filtering operations over the graph topology. The GCN update rule at layer l can be written as:

$$H^{(l)} = \sigma \left(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H^{(l-1)} W^{(l)} \right), \quad (3.5)$$

where $H^{(l)} \in \mathbb{R}^{n \times d_l}$ denotes the matrix of node embeddings at layer l with initialization $H^{(0)} = X$, the input feature matrix. The matrix $\tilde{A} = A + I \in \mathbb{R}^{n \times n}$ is the adjacency matrix augmented with self-loops, $\tilde{D} \in \mathbb{R}^{n \times n}$ is the corresponding diagonal degree matrix defined by $\tilde{D}_{vv} = \sum_u \tilde{A}(v, u)$. The matrix $W^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ contains learnable parameters, and $\sigma(\cdot)$ denotes a non-linear activation function.

The symmetric normalization term $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ ensures numerical stability and prevents feature magnitudes from growing uncontrollably as multiple layers are stacked. From a message-passing perspective, the message function corresponds to a linear transformation of neighboring node embeddings, while the aggregation function implements a degree-normalized summation over the local neighborhood. This formulation enables information to be propagated efficiently across the graph while preserving permutation invariance with respect to node ordering.

Thanks to their computational efficiency and scalability, GCNs have been successfully applied to a wide range of graph-based learning tasks, including link prediction, node and graph-level classification. Despite their empirical effectiveness, GCNs are subject to well-documented limitations. One prominent issue is over-smoothing, whereby node representations tend to converge to similar values as the number of layers increases, therefore reducing the model’s discriminative power. This phenomenon has motivated the development of alternative GNN architectures that incorporate attention mechanisms, adaptive aggregation strategies, residual connections, or normalization techniques to mitigate excessive feature homogenization. Nevertheless, GCNs remain a cornerstone of graph representation learning and serve as a foundational building block for many advanced GNN variants. In this thesis, the GNN framework - and GCNs in particular - constitutes the computational backbone for learning representations from graph-structured data derived from time series, including visibility graphs and dynamically evolving relational structures. By integrating graph-theoretic representations with deep learning models, GNN-based approaches enable the joint exploitation of temporal dynamics and network topology, providing a framework for analyzing time series data.

3.3 Signature Transform

The notion of the signature of a path originates from rough path theory and provides an expressive representation of sequential data by encoding the temporal evolution of a time series in a structured algebraic form. The strength of the signature transform lies in its ability to capture both temporal and geometric characteristics of time series data. In the context of univariate time series, temporal characteristics refer to long-range dependencies, recurring patterns, and global trends over time, whereas geometric characteristics describe the shape of the trajectory induced by suitable transformations of the time series - such as the lead-lag transformation - as well as more intricate behaviors, including loops and self-intersections (Terry Lyons, 2014).

For clarity and consistency, we adopt the notation introduced in Liao et al. (2023) and restrict our attention to continuous paths defined on a compact time interval $J := [a, b] \subset \mathbb{R}$, taking values in $\mathbb{R}^d$ and possessing finite p -variation¹. We further assume that all paths start from the origin. The space of such paths is denoted by

$$C_0^p(J, \mathbb{R}^d)$$

Let

$$\mathbb{T}(\mathbb{R}^d) := \bigoplus_{k=0}^{\infty} (\mathbb{R}^d)^{\otimes k}$$

denote the tensor algebra over $\mathbb{R}^d$. This space provides a natural codomain for the signature transform and contains the complete collection of iterated integrals associated with $\mathbb{R}^d$ -valued paths, thereby yielding a comprehensive and non-parametric representation of their evolution. Here, d denotes the dimensionality of the paths.

Finally, let $S_i = \{s_i(t_0), s_i(t_1), \dots, s_i(t_T)\}$ denote a discrete univariate time series observed at time points $t_0, t_1, \dots, t_T$. Such discrete observations can be embedded into a continuous path through appropriate

¹The p -variation quantifies the roughness or irregularity of a path. Larger values of p -variation correspond to increased path roughness. A detailed discussion can be found in Appendix A of Liao et al. (2023).

interpolation schemes, enabling the application of signature-based representations to real-world time series data. One widely used interpolation scheme is piecewise-linear interpolation, which connects consecutive observations by straight-line segments. While simple and effective, this representation may fail to capture certain temporal ordering properties when multiple time series are involved. To address this limitation, the lead-lag transformation or the time-joined transformation are often employed (D. Levin, T. Lyons, and H. Ni, 2016). In the univariate case, the lead-lag transform embeds the original time series into a two-dimensional path by separating each increment into a lead component and a lag component. This construction preserves the temporal ordering of increments and enables the signature to capture second-order and higher-order interactions that would otherwise be lost under standard interpolation. As a result, geometric features such as loops and areas enclosed by the path become explicitly represented in the signature.

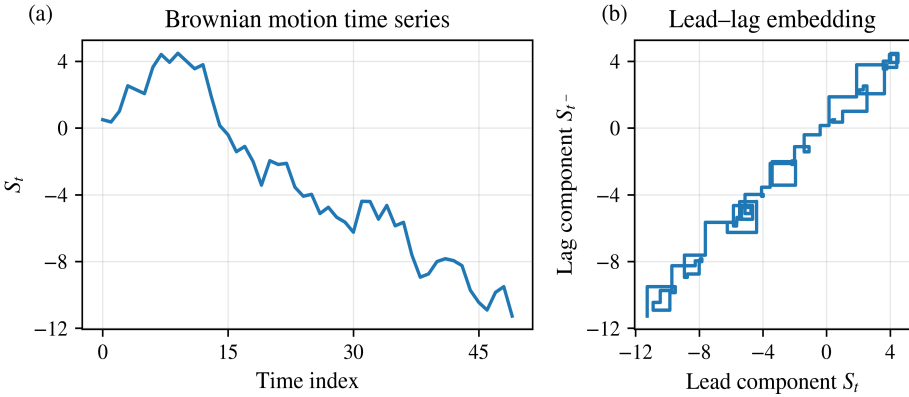


Figure 2: Lead-Lag transformation applied to a Geometric Brownian Motion with 50 observations. Panel (a) depicts the trajectory of the underlying Brownian motion, while panel (b) illustrates the corresponding lead-lag embedding of the time series.

Figure 2 illustrates the application of the lead-lag transformation to a

standard Brownian motion. Following Gyurkó et al. (2014), let $\{s_i(t_j)\}_{j=0}^T$ denote the discrete-time representation of a generic univariate time series S_i . The corresponding lead process $\{\dot{s}_z^{\text{lead}}\}_{z=0}^{2T}$ defined as

$$\dot{s}_z^{\text{lead}} = \begin{cases} s_i(t_j) & \text{if } z = 2j \\ s_i(t_j) & \text{if } z = 2j - 1 \end{cases}$$

Similarly, the lag process $\{\dot{s}_z^{\text{lag}}\}_{z=0}^{2T}$ is defined as

$$\dot{s}_z^{\text{lag}} = \begin{cases} s_i(t_j) & \text{if } z = 2j \\ s_i(t_j) & \text{if } z = 2j + 1 \end{cases}$$

The lead-lag transformation is then obtained by pairing the lead and lag processes, yielding the two-dimensional path

$$L = \{\dot{s}_z^{\text{lead}}, \dot{s}_z^{\text{lag}}\}_{z=0}^{2T} \quad (3.6)$$

The resulting path L is two-dimensional, where the first component corresponds to the lead process and the second component corresponds to the lag process.

Let L denote the continuous path produced by the lead-lag transformation², as defined in equation (3.6). Consequently, we define the signature $\mathcal{S}$ and the truncated signature at level M , denoted as $\mathcal{S}_M$, as follows:

Definition 1 (Signature and Truncated Signature) *Let $L \in C_0^p(J, \mathbb{R}^d)$ be a continuous path. The signature $\mathcal{S}$ of the path L is defined as:*

$$\mathcal{S}(L) = (1, L_J^1, \dots, L_J^k, \dots) \in \mathbb{T}(\mathbb{R}^d), \quad (3.7)$$

where

$$L_J^k = \int_{t_1 < t_2 < \dots < t_k, t_1, \dots, t_k \in J} dL_{t_1} \otimes \dots \otimes dL_{t_k}$$

are the k -th order iterated integrals³.

The truncated signature of degree M is defined as:

$$\mathcal{S}_M(L) = (1, L_J^1, \dots, L_J^M). \quad (3.8)$$

²The lead-lag transformation is adopted due to its ability to extract informative features such as path volatility, which emerges from second-order signature terms and plays a central role in financial modeling, as discussed in Remark 4.1 of D. Levin, T. Lyons, and H. Ni (2016).

³The tensor product $\otimes$ denotes all the combinations of components taken from dL_{t_1} to dL_{t_k} .

Note that L_j^k should be interpreted as a d^k -dimensional tensor rather than a scalar variable. The signature allows a hierarchical interpretation: lower-order terms capture coarse path characteristics such as total increments, while higher-order terms encode increasingly refined information, including higher-order moments and local geometric features. Crucially, the signature is invariant under time reparameterization, meaning that its values are preserved under monotone changes of time. It also satisfies important algebraic properties, including translation invariance and Chen’s concatenation identity (K.-T. Chen, 1958). The former states that the signature of a path depends only on its increments and not on its absolute position in space, meaning that the signature is able to capture the intrinsic geometric structure of a path independently of its spatial location. In the context of time series analysis, translation invariance is particularly advantageous when modeling non-stationary data, such as financial prices or sensor measurements, where absolute values may vary due to external factors, but the underlying dynamics - such as trends, volatility, and temporal dependencies - remain informative while Chen’s concatenation identity describes how the signature behaves under the concatenation of paths. In particular, this identity states that if we consider a path defined in a generic time interval $[a, b]$, and we split it in two sub-intervals $[a, u]$ and $[u, b]$ with $a < u < b$, then we compute the signature on each sub-path. If we take a tensor product of the signature of both subpaths, we obtain the signature of the whole path, hence:

$$\mathcal{S}(L)_{[a,b]} = \mathcal{S}(L)_{[a,u]} \otimes \mathcal{S}(L)_{[u,b]}$$

This property is especially valuable in time series modeling, where long sequences can be naturally segmented into shorter intervals. Using Chen’s identity, the signatures of individual segments can be multiplicatively combined to recover the signature of the entire sequence, facilitating incremental learning, and multi-scale analysis.

Note that when truncating the signature at level M , the first $\frac{d^{M+1}-1}{d-1}$ iterated integrals are retained, where d denotes the path dimension. The factorial decay of higher-order iterated integrals ensures that truncation incurs minimal information loss (Lemerrier et al., 2021).

Finally, the signature transform can be used to compare two different stochastic processes. Let A and B be two stochastic processes defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω represents the sample space, $\mathcal{F}$ is a sigma-algebra of subsets of Ω , and $\mathbb{P}$ is a probability measure (Resnick, 2019). Suppose that the signature defined in equation (3.7) exists with a high probability for both A and B , and that the expected values $\mathbb{E}[S(A)]$ and $\mathbb{E}[S(B)]$ are finite. Then, the following result holds (Terry Lyons and Hao Ni, 2015):

Theorem 1 (Expected Signature) *Let A and B be two $C_0^1(J, \mathbb{R}^d)$ -valued random variables. If*

$$\mathbb{E}[S(A)] = \mathbb{E}[S(B)]$$

and $\mathbb{E}[S(A)]$ has infinite radius of convergence, then $A \stackrel{d}{=} B$, i.e., A and B are equal in distribution.

The signature uniquely characterizes a path under suitable regularity conditions (Terry J Lyons, 1998). Moreover, the expected signature uniquely determines the distribution of a stochastic process, playing a role analogous to that of the moment generating function in classical probability theory (Chevyrev and Terry Lyons, 2016). A comprehensive treatment of the theory, along with rigorous proofs and illustrative examples, can be found in Terry Lyons (2014), D. Levin, T. Lyons, and H. Ni (2016), and Chevyrev and A. Kormilitzin (2016).

Chapter 4

Signature-Based Network Modeling for Financial Time Series

*"It is a strange fate that we should suffer so much fear and doubt
over so small a thing ... such a little thing."*

Boromir,

The Lord of The Rings: The Fellowship of the Ring

Traditional similarity measures employed in financial network analysis, such as the Pearson correlation coefficient, quantify primarily linear pairwise relationships between assets. Although computationally efficient and widely adopted, these measures disregard the temporal ordering of observations and fail to capture the higher-order interactions embedded within financial trajectories. Since financial time series are inherently nonlinear, non-stationary, and characterized by complex temporal dependencies, relying exclusively on correlation may provide an incomplete representation of asset similarity. To overcome these limitations, this chapter investigates the use of the signature transform as an alternative path representation for financial time series. Derived from rough path theory, the signature transform encodes the temporal evolution of a path through iterated integrals, preserving both sequential informa-

tion and its underlying geometric structure. Unlike correlation-based approaches, the similarity computed in the signature space compares the complete geometric characteristics of financial trajectories rather than isolated observations or linear dependencies. Consequently, signature representations have the potential to capture richer structural relationships among assets, providing a more informative basis for financial network analysis.

Building upon this motivation, this chapter presents two research contributions that exploit the signature transform for the analysis of financial time series. First, we demonstrate how signature representations can be used to construct a signature-based similarity matrix that replaces the classical correlation matrix for financial network construction. We show that this representation improves the quality of community detection, yielding higher modularity than traditional correlation-based networks (Gregnanin, De Smedt, et al., 2024). Second, we illustrate how the resulting community structure can be integrated into a network-based portfolio allocation framework, leading to improved diversification and portfolio construction (Gregnanin, Y. Zhang, et al., 2024). Together, these contributions establish a unified signature-based framework for financial network analysis, in which geometric path representations replace conventional linear similarity measures to support both community detection and portfolio optimization. This chapter is based on the published work of Gregnanin, De Smedt, et al. (2024) and Gregnanin, Y. Zhang, et al. (2024), which collectively demonstrate the effectiveness of signature-based representations to model relationships among financial assets.

4.1 Introduction

In recent years, the analysis of community structure in complex networks has attracted considerable attention across multiple scientific domains, including biological, social, communication, economic, and financial systems (Barabási and Pósfai, 2016). Community detection aims to identify groups of nodes that are more densely connected internally than externally, thereby revealing the latent organization of a network.

In the context of financial markets, communities often correspond to groups of assets that share similar dynamics or are influenced by common economic factors. Identifying such communities is of central importance for risk management, diversification, and portfolio construction (Prigent, 2007). Community detection in financial networks is, however, particularly challenging due to the temporal dependence, noise, and non-stationarity that characterize financial time series (Tsay, 2005). Conventional approaches typically transform multivariate financial time series into graph representations by computing a correlation matrix and subsequently filtering it to remove noise (Mantegna and Stanley, 1999; Sinha et al., 2010). Common filtering techniques include threshold-based methods, often referred to as the Asset Graph approach (Onnela, Kaski, and Kertész, 2004; Heimo, Kaski, and Saramäki, 2009), as well as techniques based on Random Matrix Theory (RMT) (Mehta, 2004; Z. Bai and Silverstein, 2010), which aim to discard eigenvalues associated with noise. Despite their use, correlation-based representations suffer from intrinsic limitations, most notably the assumption of linear dependence and temporal stationarity, which are frequently violated in empirical financial data (Brockwell and Davis, 2002).

The relevance of network-based representations becomes particularly evident in the context of portfolio optimization. Classical portfolio allocation models, most notably the mean–variance framework introduced by Markowitz (Markowitz, 1952), formulate portfolio selection as a bi-objective optimization problem that seeks to balance expected return and variance. Despite its foundational role in modern finance, this model is sensitive to estimation errors in both the expected return vector and the covariance matrix, and it relies on assumptions - such as normally distributed asset returns - that are frequently violated in empirical financial data¹. In response to these shortcomings, network-based portfolio optimization has emerged as a powerful alternative framework. Peralta and Zareei (2016) demonstrated that incorporating network-derived information into the Markowitz model leads to improved portfolio perfor-

¹Comprehensive discussions of these limitations are provided in M. Chung et al. (2022) and Kolm, Tütüncü, and Fabozzi (2014).

mance. Vÿrost, Lyócsa, and Baumöhl (2019) employed centrality measures to guide asset selection, while Clemente, Grassi, and Hitaj (2021) and Clemente, Grassi, and Hitaj (2022) showed that portfolios constructed from network representations outperform traditional strategies in terms of both risk and stability. More recently, network-based approaches have been extended to cryptocurrency markets (Jing and Rocha, 2023) and to portfolio selection frameworks based on mixed-integer optimization (Ricca and Scozzari, 2024). Nevertheless, despite their empirical advantages, these approaches continue to rely on covariance or correlation matrices to characterize inter-asset dependencies. As a result, they remain subject - although to a lesser extent - to limitations arising from assumptions of temporal stationarity and linear dependence. While network filtering techniques mitigate sampling noise in historical data, they do not fundamentally overcome the inherent constraints of correlation-based representations.

In this study, we introduce a paradigm shift by replacing correlation-based similarity measures with a signature-based similarity matrix derived from the signature transform² (Terry J Lyons, 1998). Unlike correlation based measures, the signature transform captures the full temporal evolution of a time series - including nonlinear and higher-order dependencies - within a structured tensor representation (Terry Lyons, 2014). By leveraging this representation, we construct financial networks that exhibit more coherent and stable community structures. Moreover, we demonstrate how the signature-based similarity matrix can be incorporated into a portfolio allocation framework to enhance diversification and reduce portfolio risk.

The remainder of this chapter is organized as follows. Section 4.2 provides a brief overview of the proposed methodology. Section 4.3 presents the results of community detection applied to the Standard & Poor’s 500 (S&P500) index. Section 4.4 analyzes the network properties derived from the signature-based similarity matrix. Finally, Section 4.5 reports

²A detailed mathematical definition of the signature transform is provided in Chapter 3.

the empirical results obtained by employing the signature-based similarity matrix within a portfolio allocation framework. Finally, Section 4.6 provides a brief summary of the main results found in this chapter.

4.2 Preliminaries

In this section, we present an overview of the construction of the signature-based similarity matrix, followed by a review of the principal methodologies employed for filtering correlation matrices. We then formally define the community detection problem and describe the key algorithms used to address it. Finally, we provide a detailed description of the portfolio optimization frameworks adopted in our analysis.

Consider a collection of N univariate time series, denoted as $\mathbf{S} = \{\mathbf{S}_1, \dots, \mathbf{S}_N\}$, where each time series $\mathbf{S}_i$ consists of observations recorded over T discrete time steps and is represented as $\mathbf{S}_i = \{s_i(1), \dots, s_i(T)\}$. Within the network-based modeling framework, a standard approach for constructing a graph from $\mathbf{S}$ consists in computing the correlation matrix among the N time series. The entries of the correlation matrix $C \in \mathbb{R}^{N \times N}$, denoted as c_{ij} , are defined as follows:

$$c_{ij} = \frac{\sigma_{S_i, S_j}}{\sqrt{\sigma_{S_i}^2} \sqrt{\sigma_{S_j}^2}}. \quad (4.1)$$

Where σ_{S_i, S_j} denotes the covariance between time series $\mathbf{S}_i$ and $\mathbf{S}_j$, and $\sigma_{S_i}^2$ denotes the variance of time series $\mathbf{S}_i$. Empirically, the covariance and variance are computed as:

$$\begin{aligned} \sigma_{S_i, S_j} &= \frac{1}{T} \sum_{t=1}^T s_i(t) s_j(t) - \frac{1}{T} \sum_{t=1}^T s_i(t) \frac{1}{T} \sum_{t=1}^T s_j(t), \\ \sigma_{S_i}^2 &= \frac{1}{T} \sum_{t=1}^T s_i^2(t) - \left(\frac{1}{T} \sum_{t=1}^T s_i(t) \right)^2. \end{aligned}$$

The resulting correlation matrix provides a pairwise measure of linear dependence between time series and serves as the basis for constructing correlation-based financial networks.

4.2.1 Signature-based Similarity Matrix

Let $d(S_i, S_j)$ denote a distance function between two time series S_i and S_j , and let $\mathbf{S}_M(L_i)$ denote the truncated signature with truncation degree M of the path L_i associated with the time series S_i . In this chapter, we assume that time series with highly similar signatures exhibit similar underlying behavior. This assumption is theoretically motivated by Theorem 1 in Chapter 3 concerning the expected signature. Formally, we compute the distance between two time series' paths in the truncated signature space, i.e.:

$$d(S_i, S_j) := d(\mathbf{S}_M(S_i), \mathbf{S}_M(S_j))$$

and express the above assumption as follows:

Assumption 1 For all $S_i, S_j \in \mathbf{S}$,

$$d(\mathbf{S}_M(S_i), \mathbf{S}_M(L_j)) \simeq 0 \implies S_i \sim S_j$$

where the symbol $\sim$ denotes similar behavior.

Consequently, the smaller the distance between the truncated signatures of two time series S_i and S_j , the more likely the corresponding time series are to exhibit similar dynamics.

We construct a signature-based similarity matrix through the following multi-step procedure:

- i) For each time series in $\mathbf{S}$ derive the associated path L by applying the lead-lag transformation.
- ii) Compute the truncated signature of degree M for each path L .
- iii) Construct a distance matrix $D \in \mathbb{R}^{N \times N}$ using a distance function $d(\cdot, \cdot)$ in truncated signature space, defined as:

$$D = \begin{bmatrix} d(\mathbf{S}_M(L_1), \mathbf{S}_M(L_1)) & \cdots & d(\mathbf{S}_M(L_1), \mathbf{S}_M(L_N)) \\ \vdots & \ddots & \vdots \\ d(\mathbf{S}_M(L_N), \mathbf{S}_M(L_1)) & \cdots & d(\mathbf{S}_M(L_N), \mathbf{S}_M(L_N)) \end{bmatrix},$$

where $d(\mathbf{S}_M(L_i), \mathbf{S}_M(L_i)) = 0$ for all $i = 1, \dots, N$, and $d(\mathbf{S}_M(L_i), \mathbf{S}_M(L_j)) \in [0, +\infty)$ for all $i \neq j$.

- iv) Transform the distance matrix D into a similarity matrix P by applying a strictly monotone decreasing function. Specifically, we define:

$$p_{ij} = \frac{1}{a + d_{ij}} \quad (4.2)$$

and, for simplicity, we set $a = 1$. The resulting similarity matrix P takes the form:

$$P = \begin{bmatrix} 1 & p_{12} & \cdots & p_{1N} \\ p_{21} & 1 & \cdots & p_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & 1 \end{bmatrix},$$

where $p_{ij} = \frac{1}{1+d(\mathbf{S}_M(L_i), \mathbf{S}_M(L_j))} \in [0, 1]$, for all $i, j \in 1, \dots, N$.

To employ the similarity matrix P as a substitute for the correlation matrix C , it is necessary to ensure that P is symmetric Positive Definite (PD) or, at least, symmetric Positive Semi-Definite (PSD). This property is established in the following proposition:

Proposition 1 *If the truncated signatures $\mathbf{S}_M(L_i)$ for all $i = 1, \dots, N$ are all distinct, then the matrix P is symmetric PD. Otherwise, it is symmetric PSD.*

Proof. The matrix P is symmetric by construction. Moreover, the transformation $p_{ij} = \frac{1}{1+d_{ij}}$ is obtained by applying the function $f(r) = \frac{1}{1+r}$, for $r = d_{ij} \geq 0$, the function f is completely monotone³. By Schoenberg's theorem (Fasshauer, 2007, Theorem 5.2), it follows that P is PD if the truncated signatures $\mathbf{S}_M(L_i)$ are all different⁴. If some truncated signatures coincide, the result follows by a limiting argument, implying that P is PSD. □

³ A function $f : [0, +\infty) \rightarrow \mathbb{R}$ is called completely monotone if $f \in C[0, +\infty) \cap C^\infty(0, +\infty)$ and $(-1)^l f^{(l)}(r) \geq 0$ for all $r > 0$ and $l = 0, 1, 2, \dots$ (Fasshauer, 2007, Definition 5.1). It is straightforward to verify that $f(r) = \frac{1}{1+r}$ satisfies this property, since $f^{(l)}(r) = (-1)^l \frac{l!}{(1+r)^{l+1}}$ (see also (Fasshauer, 2007, Example 5.3)).

⁴The same argument can be applied for the case of the transformation $p_{ij} = \frac{1}{a+d_{ij}}$ obtained by applying, for $r = d_{ij} \geq 0$, the function $f(r) = \frac{1}{a+r}$ with $a > 0$, since also that function is completely monotone (as it can be checked by reasoning in a similar way as at the end of footnote 3).

4.2.2 Filtering Method

We consider two principal approaches for filtering similarity matrices: the Asset Graph method and Random Matrix Theory (RMT). By filtering, we refer to the process of removing noisy or spurious elements from a similarity matrix while retaining only the most informative relationships among time series.

In the Asset Graph approach, all matrix entries whose values are greater than or equal to a predefined threshold are retained, while the remaining entries are discarded. The selection of this threshold is critical, as an excessively low threshold may lead to an almost fully-connected graph, whereas an excessively high threshold may result in a disconnected graph, thereby indicating either insufficient or excessive information removal. In practice, multiple threshold values are often evaluated during the filtering process (Ricca and Scozzari, 2024), or the threshold itself is treated as a hyperparameter to be optimized. We can derive the threshold based on statistical significance, following the methodology proposed in MacMahon and Garlaschelli, 2015. Assuming that the time series in $\mathbf{S}$ are independent and normally distributed, the null hypothesis (Fisher, 1915) states that the transformed random variables x_{ij} follow a normal distribution with mean zero and standard deviation $\sigma_x = (T - 3)^{-1/2}$. Each random variable x_{ij} is defined via Fisher's z -transformation as:

$$x_{ij} = \frac{1}{2} \ln \frac{1 + c_{ij}}{1 - c_{ij}} = \text{artanh } c_{ij} ,$$

where c_{ij} denotes the entries of the empirically estimated similarity matrix, for example a correlation matrix. Under this framework, the statistically significant realizations of x_{ij} are those satisfying:

$$|x_{ij}| \geq \theta \sigma_x ,$$

where θ represents a chosen significance threshold. That is, only values that lie at least θ standard deviations away from zero are considered statistically significant. Accordingly, the critical value for filtering the

correlation matrix is given by:

$$c_\theta = \frac{\exp\left(\frac{2\theta}{\sqrt{T-3}}\right) - 1}{\exp\left(\frac{2\theta}{\sqrt{T-3}}\right) + 1} = \tanh(x_\theta),$$

where $x_\theta = \theta\sigma_x$. The entries of the filtered similarity matrix, denoted as c_{ij}^* , are then defined as:

$$c_{ij}^* = \begin{cases} |c_{ij}|, & \text{if } |c_{ij}| \geq c_\theta, \\ 0, & \text{otherwise.} \end{cases} \quad (4.3)$$

An advantage of this approach is that the threshold θ can be directly derived from the critical values of the standard normal distribution⁵. A limitation of this method is the underlying assumption of normality for the transformed random variables. This assumption is often violated in financial time series, such as log-returns, which empirically exhibit well-known stylized facts, including heavy-tailed distributions relative to the Gaussian case (Cont, 2001).

An alternative filtering approach is provided by RMT, which aims to extract meaningful information from the similarity matrix by distinguishing statistically significant eigenvalue components from noise. Consider a correlation matrix computed from N independent random time series of length T . Under the asymptotic regime $N \rightarrow +\infty$, $T \rightarrow +\infty$, with $1 < \lim T/N < +\infty$, the eigenvalues of the correlation matrix follow the Marcenko-Pastur distribution (Laurent Laloux et al., 1999; Plerou, Gopikrishnan, Rosenow, Amaral, and Stanley, 1999), given by:

$$\rho(\lambda) = \begin{cases} \frac{Q}{2\pi\sigma^2} \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{\lambda} & \text{if } \lambda \in [\lambda_-, \lambda_+], \\ 0 & \text{otherwise} \end{cases} \quad (4.4)$$

and zero otherwise. Here,

$$Q = \lim \frac{T}{N}, \quad \lambda_{\pm} = \sigma^2 \left(1 \pm \sqrt{\frac{1}{Q}}\right)^2,$$

⁵Table 10 in Appendix A.1 reports the confidence levels and corresponding critical values considered in this thesis.

and σ^2 denotes the variance of the matrix elements, often estimated empirically as:

$$\sigma^2 = 1 - \frac{\lambda_{max}}{N},$$

where λ_{max} is the largest eigenvalue of the correlation matrix. Within the RMT framework, eigenvalues exceeding λ_+ are considered statistically significant, while those within the Marcenko–Pastur bounds are attributed to noise. Accordingly, any correlation matrix can be decomposed into the sum of a structural component, denoted by $C^{(s)}$, which is associated with eigenvalues exceeding the threshold λ_+ , and a noise component, denoted by $C^{(r)}$, that can be expressed as:

$$C^{(r)} = \sum_{i: \lambda_i \leq \lambda_+} \lambda_i v_i v_i^\dagger. \quad (4.5)$$

where v_i denotes the eigenvector associated with eigenvalue λ_i , and $v_i^\dagger$ its conjugate transpose.

In empirical financial data, an eigenvalue significantly larger than all others is typically observed and is commonly referred to as the market mode (Sinha et al., 2010; Mehta, 2004; Laurent Laloux et al., 1999). This dominant component reflects the collective market behavior affecting all assets. Removing the market mode is therefore essential for enhancing the detection of meaningful inter-asset correlations. The correlation matrix can thus be decomposed as:

$$C = C^{(r)} + C^{(m)} + C^{(g)}, \quad (4.6)$$

where

$$C^{(m)} = \lambda_{max} v_{max} v_{max}^\dagger, \quad (4.7)$$

$$C^{(g)} = \sum_{i: \lambda_+ < \lambda_i < \lambda_{max}} \lambda_i v_i v_i^\dagger, \quad (4.8)$$

Here, $C^{(r)}$ represents the noise component, $C^{(m)}$ the market component, and $C^{(g)}$ the remaining structurally significant correlations. In this chapter, the matrix $C^{(g)}$ is used as the filtered correlation matrix.

4.2.3 Community Detection

Community detection aims to identify groups of nodes within a network that are more densely interconnected with each other than with nodes belonging to other groups (Barabási and Pósfai, 2016; Fortunato, 2010). To identify non-overlapping communities, we employ a modularity optimization framework (Newman and Girvan, 2004). This approach is selected due to its foundation on a null model that provides a reference structure against which the observed network organization can be evaluated. Modularity serves as a quantitative measure of the quality of a network partition. Specifically, partitions characterized by high modularity exhibit dense intra-community connections and sparse inter-community connections.

Consider a network composed of N nodes, represented by an adjacency matrix $A \in \mathbb{R}^{N \times N}$, as defined in equation (3.1). The objective is to identify non-overlapping communities described by a N -dimensional vector η , where the component η_i specifies the community membership of node u , as described in (MacMahon and Garlaschelli, 2015). Recall that equation (3.2) defines the modularity function. More generally, the modularity can be expressed with respect to an arbitrary null model as:

$$Q(\eta) = \frac{1}{A_{tot}} \sum_{i,j} [A_{ij} - \langle A_{ij} \rangle] \delta(\eta_i, \eta_j), \quad (4.9)$$

where $\delta(\eta_i, \eta_j)$ is the Kronecker delta function, equal to 1 if $\eta_i = \eta_j$ and 0 otherwise, ensuring that only node pairs belonging to the same community contribute to the sum. The normalization constant $A_{tot} = \sum_{i,j} A_{ij} = 2l$ corresponds to twice the total number of links l in the network, and $\langle A_{ij} \rangle$ denotes the expected adjacency under the selected null model. A commonly adopted null model is the configuration model⁶, where $\langle A_{ij} \rangle = \frac{k_i k_j}{2l}$, with k_i representing the degree of node i (Fortunato, 2010).

When financial networks are constructed from finite time series data, the corresponding correlation matrix often exhibits a dominant global com-

⁶Used to define the modularity function in equation (3.2).

ponent, commonly referred to as the market mode. In this setting, modularity can be reformulated by incorporating the RMT decomposition of the correlation matrix. Specifically, the modularity can be expressed as:

$$\begin{aligned} Q(\eta) &= \frac{1}{C_{norm}} \sum_{i,j} \left[C_{ij} - C_{ij}^{(r)} - C_{ij}^{(m)} \right] \delta(\eta_i, \eta_j), \\ &= \frac{1}{C_{norm}} \sum_{i,j} C_{ij}^{(g)} \delta(\eta_i, \eta_j). \end{aligned} \quad (4.10)$$

where $C^{(r)}$, $C^{(m)}$, and $C^{(g)}$ denote, respectively, the noise, market, and significant correlation components⁷. The normalization constant is given by $C_{norm} = \sum_{i,j} C_{ij}$. Previous studies (Utsugi, Ino, and Oshikawa, 2004; Potters, Bouchaud, and L. Laloux, 2005; Plerou, Gopikrishnan, Rosenow, Amaral, Guhr, et al., 2002) have shown that the eigenvectors associated with $C^{(g)}$ often exhibit alternating sign structures, enabling the identification of groups of assets influenced by common latent factors. This property provides a robust basis for community detection in financial networks.

4.2.4 Asset Allocation Strategies

In this section, we provide a concise description of the network-based asset allocation strategy adopted in this study. For completeness, a detailed presentation of the classical portfolio allocation models is deferred to Appendix A.2.

In network-based portfolio strategies, the financial market is modeled as a network whose structure is derived from the correlation matrix computed from stock log-returns⁸ (Y. Li et al., 2019; Clemente, Grassi, and Hitaj, 2021; Peralta and Zareei, 2016). Let $G = (V, E)$ denote a graph representing a universe of N stocks contained in the set $\mathbf{S}$, where $V = \{v_1, \dots, v_N\}$ is the set of nodes corresponding to individual assets, and

⁷Defined in equations (4.5), (4.7), and (4.8).

⁸For any asset i , denoted as $S_i = \{s_i(1), \dots, s_i(T)\}$, we can compute the log-returns as:

$$R_i(t) = \log(s_i(t)) - \log(s_i(t-1)), \quad \forall t \in \{2, \dots, T\}.$$

$E \subseteq V \times V$ is the set of edges encoding relationships among them. Let $\mathbf{R} = \{R_1, \dots, R_N\}$ denote the set of the log-returns for each asset in $\mathbf{S}$, and let $A \in \mathbb{R}^{N \times N}$ denote the adjacency matrix associated with graph G . Note that we consider only an undirected weighted graph. The graph representation of the asset set $\mathbf{S}$ is obtained by filtering the correlation matrix C , computed from the log-returns set $\mathbf{R}$, using the Asset Graph method described in 4.2.2.

The network-based portfolio strategies used as baselines in this work are inspired by the framework proposed in Clemente, Grassi, and Hitaj (2022), with the key distinction that we consider a weighted graph rather than the unweighted formulation adopted in the original study. The underlying rationale is to jointly account for both asset volatility and the degree of interconnectedness within the financial network. To quantify the local clustering structure of the network, we compute the weighted clustering coefficient τ_i for each node i , defined as the geometric mean of the weights of triangles involving node i (Jukka-Pekka Onnela et al., 2005):

$$\tau_i = \frac{1}{k_i(k_i - 1)} \sum_{j, u \in V, i \neq j \neq u} (\hat{a}_{ij} \hat{a}_{iu} \hat{a}_{uj})^{\frac{1}{3}},$$

where k_i denotes the degree of node i , and $\hat{a}_{ij}$ represents the normalized adjacency matrix entries, computed as:

$$\hat{a}_{ij} = \frac{a_{ij}}{\max_{h,k} (a_{hk})}$$

with $a_{ij} = c_{ij}^*$. Based on the clustering coefficients, we construct the matrix C^τ , which captures the level of interconnectedness between all nodes of the network. Its entries c_{ij}^τ are defined as:

$$c_{ij}^\tau = \begin{cases} 1, & \text{if } i = j, \\ \tau_i \cdot \tau_j, & \text{otherwise.} \end{cases}$$

Finally, we define the matrix Π , which replaces the covariance matrix in the classical portfolio optimization problems⁹, analogously to the approach in Clemente, Grassi, and Hitaj (2022):

$$\Pi = \Delta^T C^\tau \Delta.$$

⁹Defined in the Appendix A.2.

Here, Δ is a diagonal matrix whose i -th diagonal entry represents the ratio between the standard deviation of the log-return of asset i , denoted as σ_i , and the overall market volatility. Specifically, the diagonal elements are given by:

$$\delta_{ii} = \frac{\sigma_i}{\sqrt{\sum_{n=1}^N \sigma_n^2}}$$

The key distinction between the matrices Π and the covariance matrix lies in the fact that Π implicitly incorporates information about the state of systemic stress of the financial market through the network structure (Clemente, Grassi, and Hitaj, 2022), while the covariance matrix accounts only for the volatility of individual assets.

4.3 Community Detection

The primary objective of this analysis is to demonstrate that replacing the traditional correlation matrix with a signature-based similarity matrix leads to a substantial improvement in modularity within the context of community detection. Higher modularity values indicate an enhanced ability to identify cohesive and well-separated communities. Moreover, our results show that the detected communities do not rigidly correspond to the predefined sector classifications, suggesting the presence of latent dependencies that exceed conventional categorizations.

To construct the signature-based similarity matrix, we follow the procedure described in Section 4.2.1. The truncation degree of the signature is set to $M = 3$. Furthermore, we consider three distinct similarity measures: Euclidean Distance (ED), Cosine Similarity (CS), and the Radial Basis Function (RBF) kernel. The selection of these measures is motivated by their complementary properties. Euclidean distance is computationally efficient and sensitive to magnitude differences between signatures; cosine similarity is scale-invariant, robust to outliers, and well suited for capturing directional trends in time series; and the RBF kernel is capable of modeling complex nonlinear relationships. Since the Euclidean distance is intrinsically a measure of dissimilarity, it is converted into a similarity metric using a strictly monotonic decreasing transformation,

as defined in equation (4.2). In contrast, cosine similarity and the RBF kernel are similarity measures by construction and therefore require no additional transformation to obtain the corresponding signature-based similarity matrix. It is important to note that the signature transform is not inherently scale invariant. If a path is rescaled by a factor c , the coefficients of the k -th signature level scale proportionally to c^k . Consequently, similarity measures computed on signatures may be influenced by differences in amplitude and volatility, even when the underlying temporal dynamics are similar. To mitigate this effect, all financial time series are converted into log-returns prior to computing their signatures. This preprocessing step removes variations arising from differences in asset price levels and enables a more meaningful comparison across assets. However, the use of log-returns does not completely eliminate scaling effects. While it removes multiplicative differences in price levels, the signature representation remains sensitive to variations in volatility because higher-order signature terms continue to scale according to the power law c^k . As a result, neither Euclidean distance nor the RBF kernel are scale invariant when applied to signature features. Similarly, cosine similarity is only partially invariant, as it is invariant to uniform rescaling of the feature vector but not to the non-uniform scaling induced across different signature levels. Therefore, the proposed framework should be interpreted as reducing, rather than fully removing, the impact of scale effects. Alternative similarity measures and embedding techniques, including Dynamic Time Warping (Berndt and Clifford, 1994), Discrete Fourier Transformation (Faloutsos, Ranganathan, and Manolopoulos, 1994), and Shapelet based methods (Ye and Keogh, 2011), were not included in the experimental comparison. The primary objective of this chapter is to evaluate whether the geometric information encoded by the signature transform provides additional discriminatory power beyond the correlation-based measures commonly employed in financial network analysis. A broader comparison against alternative time-series embeddings represents an interesting direction for future investigation.

Our empirical analysis focuses on the S&P500 index, which comprises

500 large publicly traded companies in various sectors of the United States market¹⁰. The index assigns each stock to one of eleven sectors: Utilities, Energy, Financials, Industrials, Health Care, Communication Services, Information Technology, Consumer Staples, Materials, Real Estate, and Consumer Discretionary. The dataset spans the period from Saturday 10th July, 2010 to Monday 10th July, 2023. After computing log-returns and removing assets with insufficient data, the final dataset consists of $N = 443$ stocks, each with $T = 3720$ observations.

As baseline models, we consider the S&P500 index classification partition, and the correlation-based networks filtered using both a fixed threshold and RMT-based filtering, as described in Section 4.2.2. For the threshold-based method, the optimal threshold is based on statistical significance, resulting in a retained correlation threshold of 0.0437. Under the RMT framework, the eigenvalue spectrum of the correlation matrix confirms the presence of a dominant market mode: the largest eigenvalue is approximately 174, while the second-largest eigenvalue is around 20. Consequently, the filtered correlation matrix is obtained according to equation (4.8). Notably, a comparable market mode is also observed in the signature-based similarity matrices.

Table 1 reports the modularity values obtained for both the correlation matrix and the signature-based similarity matrices, using the Louvain community detection algorithm (Blondel et al., 2008) and the Clauset-Newman-Moore greedy modularity maximization algorithm (Clauset, Newman, and C. Moore, 2004). The results demonstrate that similarity matrices derived from signatures consistently yield higher modularity values, indicating superior performance in identifying meaningful community structures compared to correlation-based approaches. The only notable exception arises when applying threshold-based filtering to cosine similarity. Overall, similarity matrices based on Euclidean distance produce the most stable and consistent results, whereas cosine similarity leads to highly variable outcomes in terms of both modularity and number of detected clusters. An additional observation is that, when applied to threshold-filtered matrices, the Louvain algorithm often identifies a

¹⁰S&P Global Homepage, <https://www.spglobal.com>.

Data Type	Filtering Method	Modularity (Louvain)	Modularity (Greedy)	# Cluster (Louvain)	# Cluster (Greedy)
S&P500 Partition	-	0.0200	0.0200	11	11
Correlation Matrix	Threshold	0.0207	0.0159	81	11
	RMT	0.0987	0.1185	4	11
Signature-based Similarity Matrix	Threshold_ED	0.1796	0.1783	8	9
	Threshold_CS	0.0020	-0.0134	430	11
	Threshold_RBF	0.0976	0.0952	112	11
	RMT_ED	0.1975	0.1994	5	11
	RMT_CS	0.1177	0.8527	2	11
	RMT_RBF	0.1326	0.1428	4	11

Table 1: Modularity results and number of clusters discovered for the correlation matrix and for the signature-based similarity matrix for the Standard and Poor’s 500.

large number of singleton communities, particularly for correlation-based and signature-based similarity matrices derived from cosine similarity and RBF kernel. This effect is largely mitigated when using Euclidean distance-based similarity matrices.

Figure 3 illustrates the community structure obtained from the Euclidean distance-based similarity matrix filtered via RMT. The algorithm identifies a total of 5 communities, in contrast to the 11 sectors defined by the S&P500 classification. Importantly, each detected community contains stocks drawn from multiple sectors, revealing hidden interdependencies and cross-sector relationships that are not captured by traditional sector-based classifications.

4.4 Network Properties

The examination of network characteristics - such as the degree distribution and the clustering coefficient - is essential for understanding the organization and topology of complex networks. Since the networks considered in this study are weighted graphs, particular attention must be given to the weighted degree (also referred to as node strength), which quantifies the total intensity of interactions associated with a given node. Formally, the weighted degree of node i is defined as the sum of the weights of all edges incident to that node. In addition, the clustering

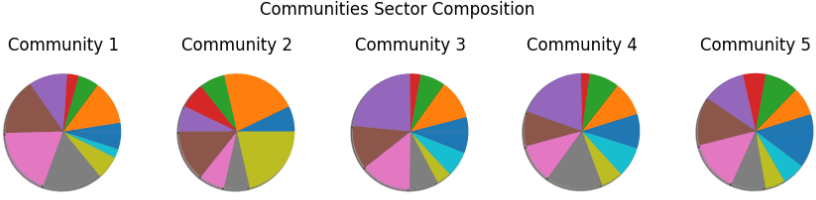


Figure 3: Communities identified using the Louvain Community Detection Algorithm on the signature-based Similarity Matrix obtained via the Euclidean Distance and subsequently filtered using RMT. Communication Services (blue), Consumer Discretionary (orange), Consumer Staples (green), Energy (red), Financials (purple), Health Care (brown), Industrials (pink), Information Technology (grey), Materials (yellow-green), Real Estate (cyan), and Utilities (blue).

coefficient measures the propensity of nodes to form tightly connected groups, thereby capturing the extent of local cohesiveness in the network.

For this analysis, we employ the same dataset used in the community detection experiments to ensure consistency and comparability of results. We focus on comparing the network derived from the signature-based similarity matrix, constructed using the Euclidean distance, with the network obtained from the conventional correlation matrix. Both matrices are filtered using the threshold-based approach, and multiple threshold values are considered in accordance with their statistical significance levels¹¹.

Figures 4 and 5 compare the distributions of node weighted degree and clustering coefficient, respectively, for graphs derived from the correlation matrix and from the signature-based similarity matrix. In both cases, the matrices are filtered using multiple threshold values. As shown in these figures, the curves corresponding to the signature-based similarity matrix (orange curves) consistently lie above those obtained from the correlation-based networks (blue curves) for both the weighted degree

¹¹Table 10 in Appendix A.1 reports the critical values used to determine the threshold in equation (4.3).

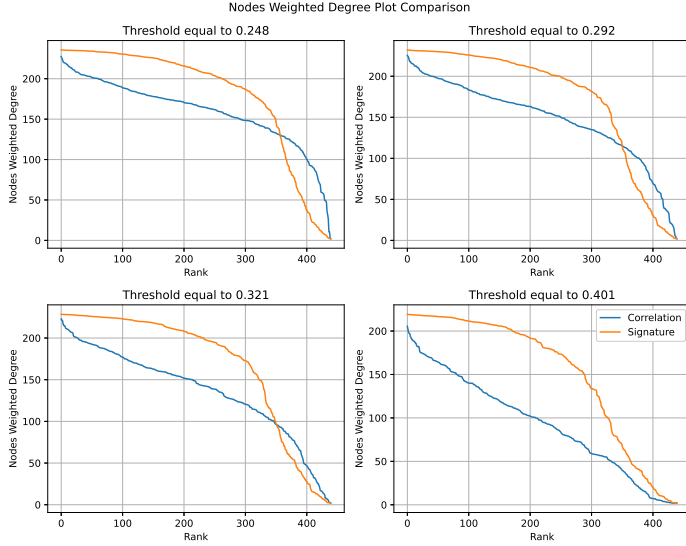


Figure 4: The plots illustrate the distribution of node weighted degree for the graphs derived from the signature-based similarity matrix (depicted by the orange curves) and from the correlation matrix (represented by the blue curves). The plots are obtained by filtering the respective matrices based on different threshold values. The term “rank” on the x-axis denotes the count of nodes possessing a particular weighted degree value.

and clustering coefficient distributions. This observation indicates that the signature-based similarity matrix captures a richer set of connectivity patterns and higher levels of local clustering than those revealed by correlation-based representations.

A further important structural property concerns the assortative or disassortative nature of the network, which characterizes whether nodes with similar properties tend to connect to each other (assortativity) or whether connections preferentially occur between dissimilar nodes (disassortativity) (Barrat et al., 2004). This behavior can be assessed by analyzing the relationship between node weighted degree and clustering coefficient. Figure 6 illustrates this relationship for both network constructions. In the graph derived from the correlation matrix (blue dots), no

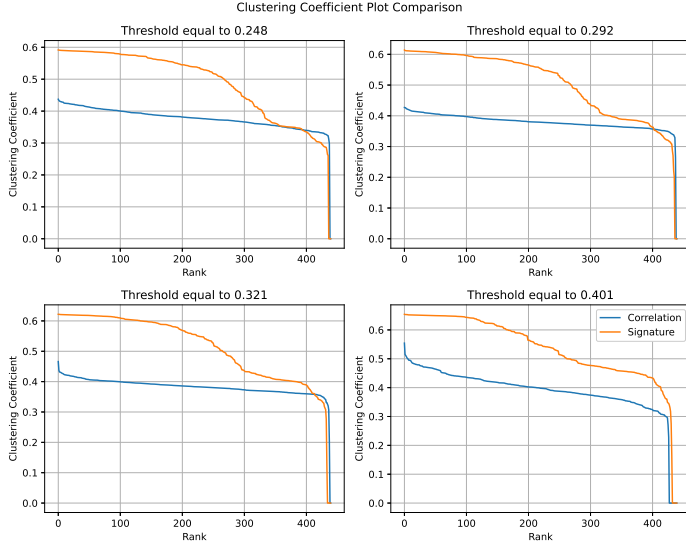


Figure 5: The plots display the distribution of node clustering coefficients for the graphs derived from the signature-based similarity matrix (illustrated by the orange curves) and from the correlation matrix (depicted by the blue curves). The plots are obtained by filtering the respective matrices based on different threshold values. The term “rank” on the x-axis denotes the count of nodes possessing a particular clustering coefficient value.

clear relationship is observed between node weighted degree and clustering coefficient. In contrast, the graph based on the signature-based similarity matrix (orange dots) exhibits a pronounced positive correlation, indicating assortative mixing and a more organized network structure.

We further examine the relationship between the volatility of each asset, measured by the standard deviation of its log-returns, and the centrality of the corresponding node in the network. Specifically, we consider degree centrality and eigenvector centrality (Barabási and Pósfai, 2016). Degree centrality reflects the number (or total weight) of connections associated with a node, whereas eigenvector centrality additionally accounts for the importance of neighboring nodes, thereby capturing influ-

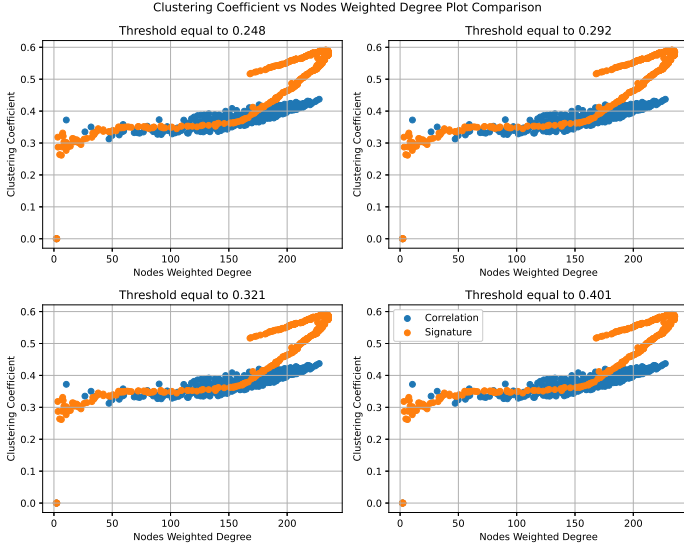


Figure 6: Comparison of the clustering coefficient and node weighted degree for the graphs derived from the signature-based similarity matrix (orange dots) and from the correlation matrix (blue dots). The plots are obtained by filtering the respective matrices based on different threshold values.

ence within the network. Figures 7 and 8 show the relationship between asset volatility and degree centrality, and between asset volatility and eigenvector centrality, respectively. For both centrality measures, a well-defined structure emerges when considering graphs derived from the signature-based similarity matrix (orange dots). This structure becomes increasingly pronounced as the underlying graph becomes sparser, corresponding to higher threshold values used to remove noisy connections. In both cases, an inverse relationship is initially observed: asset volatility decreases as centrality increases up to a certain point, after which a positive relationship emerges, indicating a transition in the role played by highly central nodes within the network.

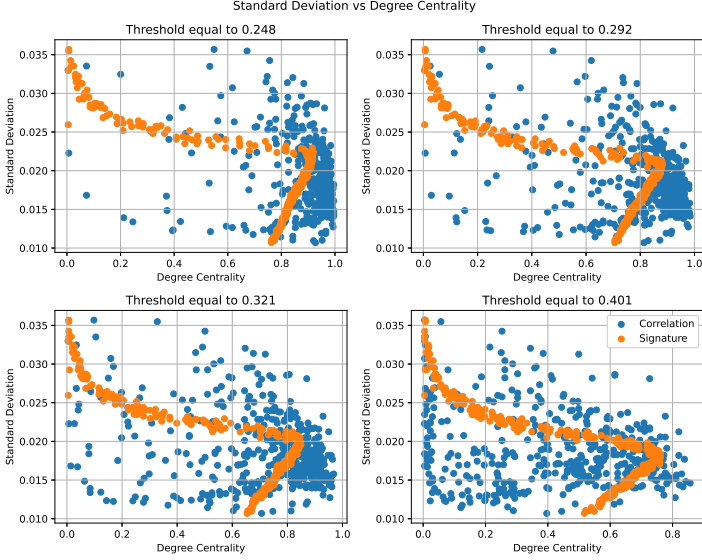


Figure 7: Comparison of the standard deviation of stocks and their degree centrality for the graphs derived from the signature-based similarity matrix (orange dots) and from the correlation matrix (blue dots). The plots are obtained by filtering the respective matrices based on different threshold values.

4.5 Asset Allocation Framework

Consistent with the community detection analysis, we focus on stocks belonging to the S&P500 index over the period from Saturday 10th July, 2010 to Monday 10th July, 2023. After removing assets with incomplete observations and computing log-returns, the resulting dataset consists of 440 stocks, each with approximately 3270 observations. The next step concerns the determination of the maximum number of assets to be included in the portfolio. Rather than imposing an *a priori* cardinality constraint (Mansini, Ogryczak, and Speranza, 2014), which arbitrarily restricts the number of selected assets, or explicitly incorporating such a constraint into the optimization problem - thereby transforming it into a mixed-integer quadratic program with substantially increased com-

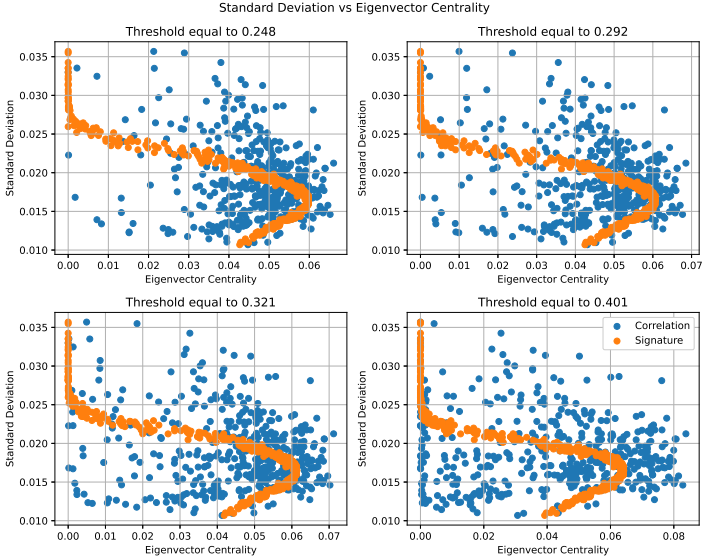


Figure 8: Comparison of the standard deviation of stocks and their eigenvector centrality for the graphs derived from the signature-based similarity matrix (orange dots) and from the correlation matrix (blue dots). The plots are obtained by filtering the respective matrices based on different threshold values.

putational complexity - we adopt a data-driven asset selection strategy based on stock turnover. Turnover, defined as the product of traded volume and price, is a widely used proxy for market liquidity and has been successfully employed in previous portfolio construction studies (Fulga, Dedu, and Șerban, 2009; Vidović, 2019). Specifically, we compute the average turnover for each stock, rank the assets accordingly, and select the top 10, 20, 40, and 80 most liquid stocks for portfolio allocation. This approach effectively mitigates liquidity-related concerns while simultaneously reducing the dimensionality and computational complexity of the optimization problem.

We compare classical portfolio strategies - Mean-Variance (MV), Global

Minimum Variance (GMV), and Maximum Sharpe Ratio (MS)¹² - using two alternative covariance proxies: a network-based implementation derived from the filtered correlation matrix, and a signature-based implementation constructed from the signature similarity matrix derived using the Euclidean distance. These approaches are denoted as *Network* and *Sig*, respectively. Performance is evaluated using the Sharpe ratio (with risk-free rate $r_f = 0$), cumulative log-returns, daily log-returns, daily standard deviation, skewness, and excess kurtosis. Portfolios are rebalanced monthly (approximately every 20 trading days), and transaction costs are assumed to be zero for simplicity.

Results for the long-only setting are reported in Tables 11–14¹³. Overall, signature-based portfolios exhibit higher daily and cumulative log-returns and lower excess kurtosis compared to network-based baselines, albeit at the cost of increased daily volatility. Sharpe ratio performance varies across asset set sizes, with no single approach dominating uniformly. Notably, portfolios composed of 10 and 20 stocks display log-return distributions close to normality, with skewness near 0 and kurtosis close to 3, underscoring the importance of risk distributional properties in portfolio construction. Across all configurations, the best cumulative returns performance is achieved by Maximum Sharpe Ratio strategies.

Equity Market Neutral Strategy

To investigate whether signature-based similarity matrices better transfer community structure into portfolio allocation, we also consider a market neutral setting by allowing short selling. The main idea of this configuration is to maintain a neutral exposure to overall market movements by balancing long and short positions. The portfolio optimization constraints¹⁴ are modified as

$$\begin{aligned} \sum_{i=1}^N w_i &= 0, \\ -1 &\leq w_i \leq 1, \forall i = 1, \dots, N, \end{aligned}$$

¹²Defined in Appendix A.2.

¹³Reported in Appendix A.3.

¹⁴Defined in Appendix A.2.

while objective functions remain unchanged. Results for this setting are reported in Tables 15–18¹⁵. For portfolios comprising 10 assets (Table 15), the comparison between signature-based and baseline network models yields mixed results, indicating that no approach consistently dominates across all evaluation metrics. In particular, signature-based portfolios achieve higher cumulative log-returns, whereas network-based portfolios attain higher daily Sharpe ratios. As the number of assets increases, a clearer trend emerges. Signature-based portfolios increasingly outperform the network baselines across nearly all metrics, with the exception of daily standard deviation, for which the signature-based approach consistently exhibits higher values. Notably, the best-performing signature-based portfolios achieve cumulative log-returns approximately 2 to 5 times larger than those of the corresponding network-based portfolios across all filtering thresholds and portfolio sizes considered. Finally, the results obtained under relaxed non-negativity constraints on portfolio weights - allowing short selling - are consistent with the findings from the community detection analysis. Overall, substituting the correlation matrix with a signature-based similarity matrix within a network-based framework yields improved portfolio performance in both long-only and market-neutral settings, albeit at the potential cost of increased volatility.

Out-of-Sample Asset Allocation

To assess the robustness of our results, we perform an out-of-sample evaluation using daily closing prices of S&P500 stocks from Tuesday 11th July, 2023 to Wednesday 31st January, 2024, yielding 141 observations per asset. Log-returns are computed, and the same set of liquid stocks selected in the in-sample analysis is retained to ensure consistency. The comparison focuses on the best-performing strategies for the network-based and signature-based approaches, denoted as *Network MS* and *Sig MS*, respectively, under both long-only and market-neutral constraints. Results are reported in Tables 19–22¹⁶. For portfolios with a small number of assets (i.e., the number of assets equal to 10), the network-based

¹⁵Reported in Appendix A.3.

¹⁶Reported in Appendix A.3.

approach outperforms the signature-based approach in the long-only setting. However, when short selling is allowed, the signature-based portfolios consistently achieve higher cumulative log-returns and Sharpe ratios, while network-based portfolios exhibit negative performance. As the number of assets increases, the signature-based approach increasingly dominates across all evaluation metrics. Overall, network-based market-neutral portfolios consistently yield negative cumulative returns and Sharpe ratios, whereas signature-based portfolios maintain positive performance in nearly all configurations, with the exception of the 40-asset case in Table 21. Moreover, the performance of the signature-based portfolios is largely stable across different threshold values.

In conclusion, the out-of-sample results confirm that signature-based portfolio construction consistently outperforms correlation-based network approaches, particularly in market-neutral settings. This suggests that signature-based similarity matrices more effectively capture higher-order and geometric dependencies in financial time series, thereby leading to more robust portfolio optimization.

4.6 Conclusion

In this chapter, we demonstrate how the signature transform can be employed to address both the community detection problem and the asset allocation task within financial markets. We begin by introducing a signature-based similarity matrix that replaces the classical correlation matrix, thereby proposing a paradigm shift in the representation of financial time series dependencies. We then describe two widely used filtering techniques for removing noise from similarity matrices, namely the Asset Graph approach and Random Matrix Theory. Finally, we present the asset allocation framework based on network representations derived from these filtered matrices.

From an empirical perspective, we first show that the signature-based similarity matrix constructed using the Euclidean distance consistently yields higher modularity values than the correlation matrix when applied to the Standard and Poor's 500 index, indicating the identifica-

tion of more coherent community structures. We subsequently analyze the structural properties of the resulting networks, comparing graphs derived from correlation-based and signature-based similarity matrices. The results demonstrate that the latter exhibits superior network characteristics, particularly in terms of clustering coefficients and node degree distributions. Lastly, we evaluate the implications of these structural differences for portfolio construction, showing that portfolios based on signature-derived similarity matrices systematically outperform those based on correlation matrices. This performance improvement is especially pronounced in the presence of short selling, where the signature-based portfolio consistently achieves superior results across all evaluation metrics considered.

Chapter 5

Time Series Forecasting using Geometric Deep Learning

"You can not hide, I see you!"

Sauron,

The Lord of The Rings: The Fellowship of the Ring

Accurate financial forecasting requires capturing both temporal dependencies and the structural relationships within financial time series. While recurrent neural networks and Transformer-based models effectively learn sequential patterns, they often overlook the geometric structure of time series, which can encode nonlinear dynamics and long-range dependencies. This chapter proposes a Geometric Deep Learning framework that represents financial time series as visibility graphs, where observations are connected based on geometric visibility. Graph Neural Networks (GNNs) exploit these graph structures to complement the temporal representations learned by conventional deep learning models. The resulting Time-Geometric framework combines temporal and geometric information to improve forecasting performance.

The chapter presents two main contributions. First, it introduces a hybrid ensemble that integrates temporal deep learning models with GNNs

to forecast the Standard and Poor’s 500 index (S&P500) (Gregnanin, De Smedt, et al., 2025b). Second, it provides a comprehensive statistical significance analysis to assess the robustness of observed improvements and the impact of geometric representations on forecast performance (Marco Gregnanin et al., 2026c). This chapter is based on one published paper, Gregnanin, De Smedt, et al. (2025b), and one manuscript under journal review, Marco Gregnanin et al. (2026c). Together, they establish a unified Time-Geometric forecasting framework and provide empirical and statistical evidence supporting the benefits of incorporating geometric information into financial time series forecasting.

5.1 Introduction

Forecasting future values of a univariate time series remains a challenging task, particularly in the financial domain, where data are characterized by stochasticity, noise, and strong nonlinear dynamics (Tsay, 2005; Lamberton and Lapeyre, 2011). Classical statistical models, such as Vector Autoregressive (VAR) models and AutoRegressive Integrated Moving Average (ARIMA) processes (Box et al., 2015), rely on assumptions of linearity and stationarity that are frequently violated in financial markets. Similarly, classical mathematical finance models based on Brownian motion and, more in general, diffusion processes (Black and Scholes, 1973; Tankov and Cont, 2003) struggle to faithfully reproduce empirical market behavior. Deep learning models have emerged as a promising alternative due to their ability to capture nonlinear temporal dependencies. Architectures such as Recurrent Neural Networks (RNNs) (Elman, 1990), Temporal Convolutional Networks (TCNs) (Lea et al., 2016), and Transformers (Vaswani et al., 2017) have demonstrated competitive performance in time series forecasting tasks (Hua et al., 2019). Nevertheless, these approaches primarily focus on temporal dependencies in Euclidean space and often neglect the underlying geometric structure of the data. Moreover, reported performance gains are frequently not accompanied by rigorous statistical significance analysis based on established evaluation criteria (Demšar, 2006).

In this chapter, we illustrate that geometric patterns play a fundamental role in financial time series forecasting. Transforming a time series from a Euclidean representation into a non-Euclidean domain via graph-based constructions enables the explicit modeling of structural properties that are otherwise inaccessible. In particular, graph representations derived from time series allow the characterization of periodic, random, and fractal behaviors through network descriptors such as degree distributions (Lacasa et al., 2008). This perspective is especially relevant in finance, where fractal and self-similar structures are pervasive (Mandelbrot, 2013). Graph Neural Networks (GNNs) (Scarselli et al., 2008) provide a natural framework for exploiting such geometric representations. Through iterative neighborhood aggregation, GNNs are able to capture multi-scale structural dependencies in a manner analogous to the self-similarity exploited in fractal analysis (W. Hamilton, Ying, and Leskovec, 2017). Building on this insight, we propose a unified forecasting framework that integrates temporal and geometric information. Specifically, univariate time series are first mapped to graph structures using the visibility graph algorithm, and dynamic GNNs are then combined with classical temporal deep learning models to jointly exploit temporal dynamics and geometric patterns. Within this framework, we introduce the *Time-Geometric* model, which couples baseline time series architectures - such as RNNs, LSTMs with attention mechanisms, TCNs, and Transformers - with dynamic GNN components. The proposed approach is evaluated extensively on financial time series data, including stocks from the S&P500 index and a broader set of 90 univariate financial series. Empirical results consistently demonstrate that incorporating geometric information leads to improved forecasting performance compared to purely temporal baselines.

Beyond predictive accuracy, this chapter places particular emphasis on statistical validation. We conduct a comprehensive statistical significance analysis, including both pairwise and multiple-comparison tests, to assess whether observed performance improvements are meaningful and robust across models and evaluation metrics. Moreover, we examine the relationship between the Mean Absolute Scaled Error (MASE) (Hyn-

dman and Koehler, 2006) and the Efficient Market Hypothesis (EMH) (Fama, 1970b), providing insight into how forecasting accuracy relates to market efficiency assumptions.

The remainder of this chapter is organized as follows. Section 5.2 presents a concise description of the proposed models and the statistical tests employed in the analysis. Section 5.3 reports the empirical results of the forecasting experiments, while Section 5.4 discusses the outcomes of the statistical significance analysis. Finally, Section 5.5 concludes the chapter.

5.2 Preliminaries

In this section, we present the main methodology adopted in this study. We begin by introducing the proposed *Time-Geometric* model and providing a detailed explanation of its architectural components. We then present a comprehensive description of the pairwise and multiple comparison statistical tests employed to assess the statistical significance of performance differences among the evaluated models.

Recalling the definition of a graph from Chapter 3, let $G = (V, E)$ denote a graph, where $V = \{v_1, \dots, v_N\}$ is the set of N nodes and $E \subseteq V \times V$ is the set of edges representing relationships between pairs of nodes. A *dynamic graph* can be defined using a snapshot-based representation (Yishu Wang et al., 2019), in which the graph evolves over discrete time steps. Formally, given a set of time indices $\mathcal{T} = \{t_1, t_2, \dots, t_T\}$ with $T = |\mathcal{T}|$, a dynamic graph is represented as a sequence of static graph snapshots:

$$\mathcal{G} = \{G_t\}_{t=1}^T,$$

where each snapshot $G_t = (V_t, E_t)$ satisfies $V_t \subseteq V$ and $E_t \subseteq E$. Both the node set and the edge set are allowed to vary across time, thereby capturing the temporal evolution of the underlying relational structure.

5.2.1 Time-Geometric model

Let $\mathbf{S}$ denote a collection of F univariate time series, each observed over T discrete time steps and represented as $S^i = \{S_1^i, \dots, S_T^i\}$, $i \in \{1, \dots, F\}$.

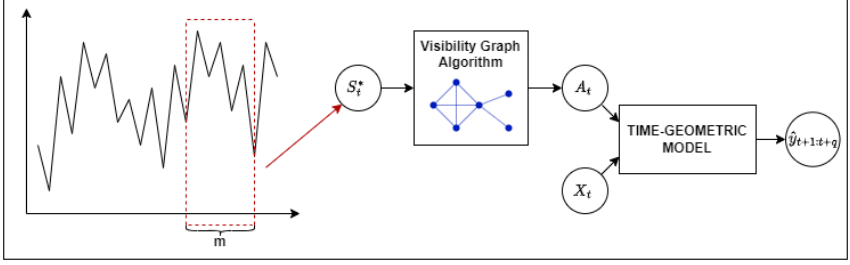


Figure 9: General idea of the proposed Time-Geometric Model.

The objective is to define a mapping function $f(\cdot)$ capable of forecasting the next q values of a target univariate time series $S^* \in \mathbf{S}$, leveraging both its graph-based representation and auxiliary information derived from related time series. Formally, the forecasting problem is expressed as learning a function f such that:

$$S_{t+1:t+q}^* = f(G_t, X_t), \quad (5.1)$$

where G_t denotes the graph representation of the target time series S^* at time t , and X_t represents an associated feature matrix. The graph G_t is constructed from the set of the most recent m observations of the target series, $S_t^* = \{S_{t-m+1}^*, \dots, S_t^*\}$, using the visibility graph algorithm. The resulting graph structure is encoded by its adjacency matrix A_t . The feature matrix X_t is formed by considering the corresponding past m observations S_t^i , $i \in \{1, \dots, F'\}$ of a subset of $F' \leq F$ time series from $\mathbf{S}$ that are correlated with the target series S^* . These auxiliary time series are represented in tabular form and serve as exogenous inputs to the model.

Figure 9 illustrates the overall conceptual framework of the proposed *Time-Geometric* model. Starting from the sliding window S_t^* , a graph-based representation is obtained via the visibility graph algorithm and encoded by the adjacency matrix A_t . The Time-Geometric model then takes A_t together with the feature matrix X_t as inputs, thereby implementing the mapping defined in equation (5.1). More precisely, the model

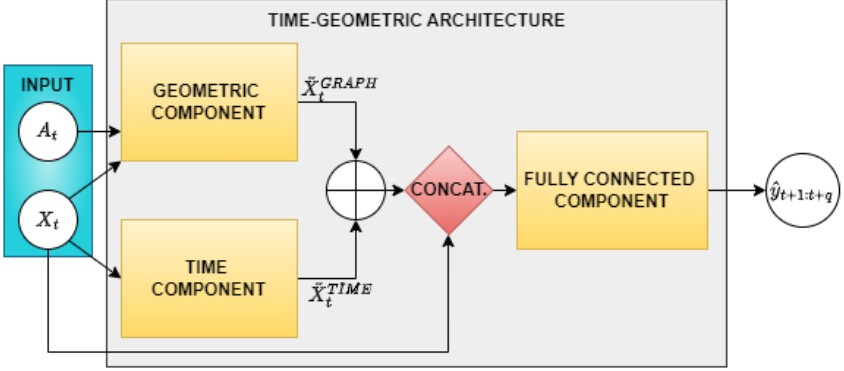


Figure 10: Time-Geometric Model Architecture.

receives as input the adjacency matrix $A_t \in \mathbb{R}^{m \times m}$ and the feature matrix $X_t \in \mathbb{R}^{m \times F'}$, where m denotes the number of past observations considered and F' is the number of auxiliary features. The model outputs a vector of predicted values $\hat{y}_{t+1:t+q} \in \mathbb{R}^q$, corresponding to the predicted feature observations of the target time series.

As shown in Figure 10, the Time-Geometric architecture is composed of three main components: a *Time Component*, which captures temporal dependencies; a *Geometric Component*, which models the structural information encoded in the graph representation; and a *Fully Connected Component*, which integrates the learned representations to produce the final forecasts.

Time Component

The Time Component takes as input the feature matrix $X_t \in \mathbb{R}^{m \times F'}$, and produces a temporal representation $\tilde{X}_t^{TIME} \in \mathbb{R}^{m \times F'}$. Its primary objective is to extract and model temporal dependencies present in the input data.

The operational steps of this component are summarized in Algorithm 1. The procedure begins by initializing the hidden representation as

Algorithm 1: Time Component

Input: Feature matrix X_t

Output: Temporal representation $\tilde{X}_t^{\text{TIME}}$

$R_t^{(0)} \leftarrow X_t;$

for $l \leftarrow 1$ **to** L **do**

$R_t^{(l)} \leftarrow \text{TIME}^{(l)}(R_t^{(l-1)});$

$\tilde{X}_t^{\text{TIME}} \leftarrow \text{FC}(R_t^{(L)});$

$$R_t^{(0)} = X_t.$$

Subsequently, a stack of L temporal neural network layers is applied. At the l -th layer, the hidden representation is updated according to

$$R_t^{(l)} = \text{TIME}^{(l)}(R_t^{(l-1)}),$$

where $\text{TIME}^{(l)}(\cdot)$ denotes a generic temporal modeling function. The Time Component may instantiate $\text{TIME}^{(l)}(\cdot)$ using one of several established temporal architectures, including LSTM networks (Hochreiter and Schmidhuber, 1997), Gated Recurrent Unit (GRU) (J. Chung et al., 2014), vanilla RNNs (Goodfellow, Bengio, and Courville, 2016), Bidirectional RNNs (BiRNNs) (Schuster and Paliwal, 1997), Bidirectional LSTM networks (BiLSTM) (Graves, Fernández, and Schmidhuber, 2005), Bidirectional GRU networks (BiGRU) (Xiong, Merity, and Socher, 2016), Transformers (Vaswani et al., 2017), and TCNs (Lea et al., 2016).

To obtain the final temporal representation, a fully connected layer is applied to the last hidden state:

$$\tilde{X}_t^{\text{TIME}} = \phi \left(W_t R_t^{(L)} + b_t \right), \quad (5.2)$$

where W_t denotes the weight matrix, b_t is a bias vector, and $\phi(\cdot)$ represents a nonlinear activation function.

Geometric Component

The Geometric Component is designed to capture structural and geometric patterns derived from the graph-based representation of the tar-

Algorithm 2: Geometric Component

Input: Adjacency matrix A_t , feature matrix X_t

Output: Geometric representation $\tilde{X}_t^{\text{GRAPH}}$

$H_t^{(0)} \leftarrow X_t$;

for $l \leftarrow 1$ **to** L' **do**

$H_t^{(l)} \leftarrow \text{GNN}^{(l)}(H_t^{(l-1)}, A_t)$;

$\tilde{X}_t^{\text{GRAPH}} \leftarrow \text{LSTM}(H_t^{(L')})$;

get time series. Its inputs consist of the adjacency matrix $A_t \in \mathbb{R}^{m \times m}$, obtained by applying the visibility graph algorithm to the window $S_t^* = \{S_{t-m+1}^*, \dots, S_t^*\}$, and the feature matrix $X_t \in \mathbb{R}^{m \times F'}$. The output is a geometric representation $\tilde{X}_t^{\text{GRAPH}} \in \mathbb{R}^{m \times F'}$. The sequence of operations is described in Algorithm 2. The component initializes the node embeddings as:

$$H_t^{(0)} = X_t.$$

A stack of L' GNN layers is then applied, yielding

$$H_t^{(l)} = \text{GNN}^{(l)}(H_t^{(l-1)}), \quad l = 1, \dots, L'$$

Each GNN layer follows the message-passing paradigm defined in equation (3.4). To incorporate temporal dependencies in the learned geometric embeddings, the output of the final GNN layer $H_t^{(L')}$ is processed by an LSTM layer, producing the geometric representation $\tilde{X}_t^{\text{GRAPH}}$.

Fully Connected Component

The Fully Connected Component integrates temporal and geometric information to produce the final forecast. Its inputs are the temporal representation $\tilde{X}_t^{\text{TIME}} \in \mathbb{R}^{m \times F'}$, the geometric representation $\tilde{X}_t^{\text{GRAPH}} \in \mathbb{R}^{m \times F'}$, and the original feature matrix $X_t \in \mathbb{R}^{m \times F'}$. The corresponding operations are summarized in Algorithm 3. First, the temporal and geometric representations are combined via element-wise addition:

$$\tilde{X}_t^{\text{TOT}} = \tilde{X}_t^{\text{TIME}} + \tilde{X}_t^{\text{GRAPH}}.$$

Algorithm 3: Fully Connected Component

Input: Temporal representation $\tilde{X}_t^{\text{TIME}}$, geometric representation $\tilde{X}_t^{\text{GRAPH}}$, feature matrix X_t
Output: Prediction $\hat{y}_{t+1:t+q}$
 $\tilde{X}_t^{\text{TOT}} \leftarrow \tilde{X}_t^{\text{TIME}} + \tilde{X}_t^{\text{GRAPH}};$
if *Skip Layer* **then**
 $\tilde{X}_t^{\text{TOT}} \leftarrow \text{CONCAT}(\tilde{X}_t^{\text{TOT}}, X_t);$
 $Z_t^{(0)} \leftarrow \tilde{X}_t^{\text{TOT}};$
for $l \leftarrow 1$ **to** L'' **do**
 $Z_t^{(l)} \leftarrow \phi(\text{FC}^{(l)}(Z_t^{(l-1)}));$
 $\hat{y}_{t+1:t+q} \leftarrow Z_t^{(L'')};$

If a skip connection is enabled, the combined representation is concatenated with the original input features:

$$\tilde{X}_t^{\text{TOT}} = \text{CONCAT}(\tilde{X}_t^{\text{TOT}}, X_t).$$

The resulting representation initializes the hidden state

$$Z_t^{(0)} = \tilde{X}_t^{\text{TOT}}.$$

A sequence of L'' fully connected layers is then applied to obtain the final prediction vector $\hat{y}_{t+1:t+q}$ that represents the forecast over the next q time steps.

5.2.2 Statistical Tests

Let K denote the number of algorithms under comparison and N_D the number of datasets. Let v_u^j denote the value of a given evaluation metric achieved by the j -th algorithm on the u -th dataset. For all statistical tests, the significance level is fixed at $\alpha = 0.05$, corresponding to a 5% probability of committing a Type I error, that is, incorrectly rejecting the null hypothesis when it is true.

Pairwise Statistical Comparison

Let $v_u^{j(BL)}$ denote the evaluation metric obtained by the baseline algorithm $j(BL)$ on the u -th dataset, and let $v_u^{j(TG)}$ denote the corresponding metric obtained by its Time-Geometric counterpart $j(TG)$. The null hypothesis for the pairwise comparison is formulated as

$$\mathbf{H}_0 : v_u^{j(BL)} = v_u^{j(TG)} . \quad (5.3)$$

which states that the baseline and its Time-Geometric extension exhibit equivalent performance. To test this hypothesis, we employ three complementary statistical tests: the Paired t -test, the Wilcoxon signed-rank test (Wilcoxon, 1992), and the Sign test (Salzberg, 1997; Sheskin, 2003).

Paired t -Test. The Paired t -test evaluates whether the mean difference in performance between the two algorithms differs significantly from zero. Let

$$d_u = v_u^{j(BL)} - v_u^{j(TG)}$$

denote the performance difference on dataset u , and let

$$\bar{d} = \frac{1}{N_D} \sum_{u=1}^{N_D} d_u$$

be the sample mean of these differences. Under the null hypothesis $\mathbf{H}_0 : \mu_d = 0$, where μ_d denotes the true mean difference, the test statistic is computed as

$$z_t = \frac{\bar{d}}{\sigma_{\bar{d}}} , \quad (5.4)$$

where $\sigma_{\bar{d}}$ is the standard deviation of the sample mean. The statistic z^t follows the Student's t -distribution with $N_D - 1$ degrees of freedom. This test provides a powerful benchmark for detecting consistent mean improvements, but its validity depends on distributional assumptions that may not always hold in practice.

Wilcoxon Signed-Rank Test. The Wilcoxon signed-rank test is a non-parametric alternative to the Paired t -test and does not assume normality

of the differences. It ranks the absolute values of the differences d_u while accounting for their signs. Let B^+ denote the sum of ranks for datasets where $j(TG)$ outperforms $j(BL)$, i.e., $v_u^{j(TG)} < v_u^{j(BL)}$, and let B^- denote the sum of ranks for the opposite case. The Wilcoxon test statistic is defined as

$$z^w = \frac{B^* - \frac{1}{4}N_D(N_D + 1)}{\sqrt{\frac{1}{24}N_D(N_D + 1)(2N_D + 1)}}, \quad (5.5)$$

where $B^* = \min(B^+, B^-)$. For sufficiently large N_D , z^w roughly follows a standard normal distribution.

By ranking the magnitudes of the differences and incorporating their signs, this test is robust to outliers and skewed distributions. Its inclusion ensures that observed performance gains are not driven by a small number of extreme cases but are instead representative across datasets.

Sign Test. The Sign test provides a minimal-assumption perspective by considering only the direction of performance differences. Let $|\#WIN|$ denote the number of datasets for which $j(TG)$ outperforms $j(BL)$. Under the null hypothesis, both algorithms are expected to outperform each other with equal probability, yielding

$$\mathbf{H}_0 : \quad \mathbb{P}\left(v_u^{j(TG)} < v_u^{j(BL)}\right) = 0.5.$$

For $N_D > 25$, the random variable $|\#WIN|$ is approximately normally distributed with mean $\frac{N_D}{2}$ and standard deviation $\frac{\sqrt{N_D}}{2}$. The performance difference is deemed statistically significant at the 95% confidence level if

$$|\#WIN| \geq \frac{N_D}{2} + 1.96 \frac{\sqrt{N_D}}{2}. \quad (5.6)$$

The Sign test is particularly valuable for confirming whether performance gains are consistently observed across datasets, even when improvements are small or uneven.

Multiple Comparison Analysis

While pairwise tests provide insight into individual model improvements, performing multiple pairwise comparisons increases the risk of Type I

errors. To control the family-wise error rate (Demšar, 2006), we additionally consider a multiple comparison analysis.

Specifically, we employ the Friedman test (Friedman, 1937; Friedman, 1940), which compares all K algorithms simultaneously for a given evaluation metric. For each dataset u , algorithms are ranked in ascending order of performance, with 1 assigned to the best-performing algorithm. Let RK_u^j denote the rank of algorithm j on dataset u , and let

$$\overline{RK}_j = \frac{1}{N_D} \sum_{u=1}^{N_D} RK_u^j$$

denote its average rank. The Friedman test evaluates the null hypothesis that all algorithms perform equivalently. The test statistic is given by

$$\chi_F^2 = \frac{12N_D}{K(K+1)} \left[\sum_{j=1}^K \overline{RK}_j^2 - \frac{K(K+1)^2}{4} \right], \quad (5.7)$$

which follows a Chi-squared distribution with $K - 1$ degrees of freedom when $N_D > 10$ and $K > 5$.

Once the Friedman test rejects the null hypothesis of equal performance, the Nemenyi test (Nemenyi, 1963) is applied to identify which specific pairs of algorithms differ significantly. Two algorithms are considered significantly different if the absolute difference between their average ranks exceeds the critical difference

$$CD = q_\alpha \sqrt{\frac{K(K+1)}{6N}}, \quad (5.8)$$

where q_α is obtained from the Studentized range distribution and scaled by $\sqrt{2}$ (Demšar, 2006). This enables an interpretable, statistically controlled comparison of all models, providing clear insight into where an algorithm outperforms its competitors.

5.3 Time Series Forecasting

The primary objective of this empirical study is to quantify the extent to which the proposed Time-Geometric model improves forecasting accuracy relative to the baseline models considered.

The empirical evaluation is conducted on the Standard and Poor’s 500 (S&P500) index, selected due to its high liquidity and its widely documented role as a proxy for the overall performance of the U.S. equity market (Chordia, Roll, and Subrahmanyam, 2001; Amihud, 2002; Welch and Goyal, 2008). We collect daily *Close*, *Open*, *High*, and *Low* price observations from Yahoo Finance¹ from Monday 4th January, 2010 to Sunday 4th June, 2023, for a total of 3,377 observations.

5.3.1 Model Specification

Figure 11 depicts the specific instantiation of the proposed Time-Geometric model used to forecast the S&P 500. The architecture comprises three components: a *Time Component*, a *Geometric Component*, and a *Fully Connected Component*.

The Time Component receives the feature matrix X_t as input and applies a stack of two LSTM layers to model temporal dependencies and long-range patterns. Dropout regularization is inserted between the LSTM layers to mitigate overfitting, and an additional dropout layer is applied prior to the attention mechanism. The attention layer assigns data dependent weights to the latent temporal features, thereby emphasizing informative components and modeling inter-feature dependencies. The attended representation is then processed by an additional LSTM layer followed by dropout, yielding the output of the Time Component, denoted by $\tilde{X}_t^{TIME} \in \mathbb{R}^{m \times F'}$.

The Geometric Component takes as input both the feature matrix X_t and the adjacency matrix A_t . Two GNN layers are applied with the hyperbolic tangent (tanh) activation function to propagate information over the graph and extract geometric patterns induced by the transformation from time-series-to-graph. The resulting node-wise representations are subsequently passed through an LSTM layer to capture sequential structure in the learned geometric embeddings. The output of this block is denoted by $\tilde{X}_t^{GRAPH} \in \mathbb{R}^{m \times F'}$.

Different choices of message and aggregation functions in Equation (3.4)

¹<https://finance.yahoo.com>

yield different GNN architectures. In this study, we consider the following three graph-convolution operators:

- **Graph Convolutional Networks (GCNs)** (Kipf and Welling, 2017). The l -th layer update is

$$H_t^{(l)} = \rho \left(\tilde{D}_t^{-1/2} \tilde{A}_t \tilde{D}_t^{-1/2} H_t^{(l-1)} \Theta \right),$$

where $\rho(\cdot)$ is an activation function, Θ is a learnable weight matrix, $\tilde{A}_t = A_t + I$ is the adjacency matrix augmented with self-loops, and $\tilde{D}_t$ is the corresponding degree matrix.

- **Chebyshev Spectral Graph Convolution (ChebConv)** (Defferrard, Bresson, and Vandergheynst, 2016). The layer update is

$$H_t = \rho \left(\sum_{r=1}^R Z_t^{(r)} \Theta^{(r)} \right),$$

where R is the Chebyshev filter size, $\Theta^{(r)}$ are learnable parameters, and $Z_t^{(r)}$ is computed recursively as $Z_t^{(1)} = X_t$, $Z_t^{(2)} = \hat{L}_t X_t$, and $Z_t^{(k)} = 2\hat{L}_t Z_t^{(r-1)} - Z_t^{(r-2)}$ for $r \geq 3$. Here, $\hat{L}_t$ denotes the rescaled Laplacian matrix at time t $\hat{L}_t = \frac{2L_t}{\lambda_{\max}} - I_t$, where L_t is the graph Laplacian and $\lambda_{\max}$ is the largest eigenvalue of L_t .

- **GraphSAGE Convolution** (W. Hamilton, Ying, and Leskovec, 2017).. The activation for node i is given by

$$h_t^i = \rho \left(\Theta_1 \mathbf{x}_t^i + \Theta_2 \text{mean}_{j \in N(i)} \mathbf{x}_t^j \right),$$

where Θ_1 and Θ_2 are learnable weight matrices and $N(i)$ is the neighborhood of node i .

Finally, the Fully Connected Component fuses the temporal and geometric representations with the original features to produce the forecast. Concretely, it combines $\tilde{X}_t^{TIME}$, $\tilde{X}_t^{GRAPH}$, and X_t , and then applies three fully connected layers with 128, 64, and q units, respectively. The Rectified Linear Unit (ReLU) activation function, $g(x) = \max(x, 0)$, is employed in the hidden layers to model the residual nonlinear relationships required for the final prediction.

5.3.2 Experiments Configuration

Considering Equation (5.1), we reformulate the forecasting task as a one-step-ahead regression problem by setting the prediction horizon to $q = 1$ and the input window length to $m = 20$. The choice of a 20-day window is motivated by the fact that the average number of trading days in a month in the United States' financial markets is approximately 21. Under this setting, the prediction function is given by

$$\hat{s}_{t+1} = f(G_t, X_t),$$

where $X_t \in \mathbb{R}^{20 \times 4}$ denotes the feature matrix composed of four variables, namely *Close*, *Open*, *High*, and *Low* prices, and G_t represents the graph constructed from the closing prices over the past 20 observations, i.e., $S_t = \{s_{t-19}, \dots, s_t\}$. The objective is to predict the closing price at time $t + 1$.

Pre-Processing

To effectively capture nonlinear patterns in the data, all input features are normalized prior to model training. Specifically, we apply the following normalization scheme:

$$\tilde{x}_t^i = \frac{x_t^i - \mu_t^i}{\sigma_t^i}, \quad (5.9)$$

where $i \in \{\textit{Close}, \textit{Open}, \textit{High}, \textit{Low}\}$, x_t^i denotes the value of the i -th feature at time t , and μ_t^i and σ_t^i are the corresponding mean and standard deviation computed over a sliding window of 20 days.

Then, the dataset is partitioned into training, validation, and test sets using a 70%/10%/20% split, respectively. The validation set is employed to implement an early stopping criterion during training, with a patience of 20 epochs, in order to mitigate overfitting.

Evaluation Metrics

To compare the predictive performance of the considered models, we adopt several standard evaluation metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean

Absolute Percentage Error (MAPE), and Mean Absolute Scaled Error (MASE) (Hyndman and Koehler, 2006). The MASE metric is defined as follows:

$$\text{MASE} = \frac{\frac{1}{l} \sum_{i=1}^l |y_i - \hat{y}_i|}{\frac{1}{l-1} \sum_{i=2}^l |y_i - y_{i-1}|}, \quad (5.10)$$

where l denotes the number of samples in the corresponding dataset split, y_i is the true target value, and $\hat{y}_i$ is the model prediction. The other evaluation metrics are defined in the Appendix B.1.

Among these metrics, MAPE and MASE are scale-free. In particular, MASE compares the model forecast against a naive benchmark in which the one-step-ahead prediction equals the most recent observation. For all metrics, lower values indicate better predictive performance. A MASE value below 1 implies that the model outperforms the naive forecast, whereas values greater than 1 indicate inferior performance. The MASE metric is especially informative in the context of the Efficient Market Hypothesis (EMH) (Fama, 1970a). According to the EMH, future stock prices cannot be consistently predicted beyond the information contained in current prices; consequently, the naive forecast is optimal. Under this hypothesis, a model should yield MASE values close to or exceeding 1.

Hyperparameters Setting and Baseline Models

Model training is performed by minimizing the Mean Squared Error (MSE) loss augmented with an ℓ_2 regularization term to reduce overfitting. Additional hyperparameters include the batch size, number of training epochs, optimization algorithm, and learning rate. In our experiments, we fix the batch size to 25, the maximum number of epochs to 500, and employ the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 10^{-4} . A dropout rate of 0.1 is applied uniformly across all models. These hyperparameters are kept constant for all architectures to ensure a fair comparison.

For benchmarking purposes, we consider the following baseline models: a vanilla GCN augmented with an LSTM layer, a BiLSTM model, an attention-based LSTM (AT-LSTM), and an ARIMA model. The GCN-

LSTM baseline consists of two GCN layers with 16 and 10 neurons, respectively, followed by an LSTM layer with 100 neurons. The BiLSTM architecture comprises three layers with 50 neurons each. The AT-LSTM mirrors the Time component shown in Figure 11, without dropout layers. Finally, the ARIMA model is specified as $\text{ARIMA}(3, 0, 2)$, where the autoregressive, differencing, and moving average orders are selected via the Akaike Information Criterion (AIC) (Akaike, 1998) over a range of candidate configurations. Recent foundation models and specialized forecasting architectures, such as PatchTST (Nie et al., 2023), TimesNet (H. Wu et al., 2023), TimeMixer (Shiyu Wang et al., 2024), TiDE (Das et al., 2024), Chronos (Ansari et al., 2024), and N-BEATS (Oreshkin et al., 2020), were not included in the experimental evaluation. The primary objective of this chapter is not to establish a new state-of-the-art forecasting architecture, but rather to evaluate whether incorporating geometric information through graph representations consistently improves predictive performance across several temporal models. A comprehensive comparison with recent time-series foundation models remains an important direction for future research.

5.3.3 Numerical Results

Table 2 reports the predictive performance of all considered models on the test set, as measured by MSE, RMSE, MAE, MAPE, and MASE. For all metrics, lower values indicate superior forecasting accuracy. We denote our proposed approach as *TA-DGNN*, where the prefix *TA* refers to the temporal attention mechanism employed in the Time Component, and *DGNN* denotes the Dynamic Graph Neural Network forming the Geometric Component of the model. As shown in Table 2, the proposed TA-DGNN framework consistently outperforms all baseline models across every evaluation metric. The strongest performance is achieved by the TA-DGNN variant based on GCN layers. A particularly noteworthy result concerns the MASE metric: with the exception of the AT-LSTM baseline, all competing models yield MASE values greater than or equal to 1, indicating that they do not consistently outperform the naive one-step-

Table 2: Model performance in terms of MSE, RMSE, MAE, MAPE and MASE for the test set. The best value for each metric is in bold.

Models	MSE	RMSE	MAE	MAPE	MASE
ARIMA(3, 0, 2)	2269.7678	47.64208	36.0789	0.8886 %	1.0137
GNN+LSTM	4742.4291	68.8653	55.5719	1.3396 %	1.5624
BiLSTM	9361.4256	96.7545	78.4831	1.9301 %	2.2066
AT-LSTM	1254.2723	35.4157	22.3470	0.5547 %	0.6276
TA-DGNN (GCN)	123.4954	11.1128	7.6060	0.1881 %	0.2136
TA-DGNN (ChebConv)	216.5798	14.7166	9.5463	0.2371 %	0.2681
TA-DGNN (SAGE)	127.2517	11.2806	7.7775	0.1925 %	0.2184

ahead forecast. In contrast, all TA-DGNN variants achieve MASE values well below 1, suggesting that they provide forecasts that systematically outperform the naive benchmark. This empirical evidence demonstrates that the proposed framework captures predictive structure in financial time series beyond that exploited by the considered benchmark models. Overall, these results demonstrate that the TA-DGNN architecture, and in particular its GCN-based instantiation, delivers substantial improvements over classical statistical models and purely temporal deep learning approaches. The consistently low MASE values further suggest that the EMH is not supported when predictions are generated using the proposed Time–Geometric modeling paradigm.

To qualitatively illustrate the differences in forecasting performance, Figure 12 compares the predicted closing prices produced by the various models against the true S&P500 closing prices over a selected interval of the test set. A closer alignment between the predicted and true trajectories indicates improved predictive accuracy. For clarity, only a limited subset of the test period is displayed.

We further assess the robustness of the proposed approach through an out-of-sample evaluation conducted under adverse market conditions. Specifically, we compare the TA-DGNN model against the AT-LSTM baseline, which achieved the strongest performance among the baselines in the in-sample analysis. This experiment focuses on a bearish market period characterized by economic stress, corresponding to the United

Table 3: Model performance in terms of MSE, RMSE, MAE, MAPE and MASE for the out-of-sample analysis. The best value for each metric is in bold.

Models	MSE	RMSE	MAE	MAPE	MASE
AT-LSTM	28.3803	5.3273	4.2145	0.4168 %	0.4682
TA-DGNN (GCN)	5.5232	2.3501	1.7421	0.1715 %	0.1935
TA-DGNN (ChebConv)	7.8564	2.8029	1.9653	0.1949 %	0.2183
TA-DGNN (SAGE)	5.7383	2.3955	1.7336	0.1710 %	0.1926

States’ financial crisis in 2008 following the mortgage crisis (Adrian and Shin, 2010). To this end, we collect daily *Close*, *Open*, *High*, and *Low* prices of the S&P500 from Yahoo Finance over the interval from Monday 1st October, 2007 to Tuesday 1st December, 2009, yielding a total of 547 observations.

The data are preprocessed following the same procedure adopted in the previous experiments, and the forecasting horizon is again set to one day. The quantitative results of this out-of-sample evaluation are reported in Table 3, while Figure 13 illustrates a visual comparison between the predicted and true closing prices over a representative subset of the test period. As shown in Table 3, the TA-DGNN model continues to outperform the AT-LSTM baseline across all evaluation metrics, confirming its robustness under challenging market conditions. While the TA-DGNN variant employing GCN layers achieves the lowest MSE and RMSE values, the variant based on GraphSAGE layers attains the best performance in terms of MAE, MAPE, and MASE. These results highlight the flexibility of the proposed framework and indicate that different GNN architectures may be advantageous depending on the specific error criterion of interest.

In summary, both the in-sample and out-of-sample analyses provide strong empirical evidence that the proposed TA-DGNN framework offers a substantial and robust improvement over existing baseline models for financial time series forecasting.

5.4 Statistical Analysis

The primary objective of the following empirical studies is to demonstrate the statistical significance of incorporating geometric patterns into the forecasting of financial time series. In addition, we investigate how the outcomes of statistical significance analyses depend on the choice of evaluation metric adopted in the assessment.

5.4.1 Problem Formulation

Let $\mathbf{S}$ denote a collection of F univariate time series, each with T observations, and X_t be the associated feature matrix. Let X_t denote the associated feature matrix. We define a baseline mapping function $g(\cdot)$ that predicts the future q values of a target time series S^* using only the information contained in X_t , namely,

$$S_{t+1:t+q}^* = g(X_t). \quad (5.11)$$

The subsequent step consists of evaluating the performance of K learning algorithms, derived from the functions $f(\cdot)$ and $g(\cdot)$, across N_D distinct datasets. Let v_u^j denote the value of a given evaluation metric obtained by the j -th algorithm on the u -th dataset. The objective is to determine whether the observed differences in evaluation metrics among the algorithms are statistically significant, considering both pairwise and multiple-comparison settings.

To this end, the analysis is conducted under the following assumptions:

Assumption 2 (Evaluation metrics comparison) *A decrease in the value of the evaluation metric from v_u^1 to v_u^2 indicates that algorithm $j = 2$ outperforms algorithm $j = 1$.*

Assumption 3 (Reliability) *The evaluation metrics v_u^j are assumed to be reliable across all K algorithms and all N_D datasets; that is, for each dataset, the metrics are computed using the same underlying data splits or random samples for all algorithms.*

5.4.2 Experiments Configuration

Dataset and Pre-Processing

For our empirical analysis, we focus on the constituents of the Standard & Poor’s 100 (S&P100) index, which comprises the largest-capitalization companies within the S&P500. We collect daily observations of *Close*, *Open*, *High*, *Low*, and *Volume* for each stock from Yahoo Finance over the period spanning from Sunday 4th January, 2015 to Sunday 4th June, 2023. After removing stocks with missing values, the resulting dataset consists of 90 univariate time series, each characterized by 5 features and approximately 2,117 observations.

Prior to normalization, the dataset is partitioned into training, validation, and test sets according to a 0.6–0.2–0.2 split. Each feature is then normalized using a rolling-window approach to obtain zero mean and unit variance, as defined in Equation (5.9).

To facilitate a comparative evaluation across models, we consider the following performance metrics: RMSE, MAE, MAPE², and MASE³.

Hyperparameters Setting and Baseline Models

Efficient model training requires the specification of appropriate hyperparameters and an optimization strategy. The baseline models correspond to the neural architectures employed in the Time Component - namely LSTM, RNN, GRU, BiLSTM, BiRNN, BiGRU, Transformers, and TCNs - and are formalized by the mapping function in Equation (5.11). The Time-Geometric model augments these baseline architectures with a GCN, as defined in Equation (5.1).

The hyperparameters subject to optimization include the batch size, the sequence length (m), and the inclusion of dropout layers and skip connections. For both the Time and Geometric Components, we further optimize the number of layers (L and L' , respectively), the number of neurons per layer, the dropout rate, and the activation functions $\phi(\cdot)$ and $\rho(\cdot)$. In addition, the Geometric Component requires the selection of the

²Defined in Appendix B.1.

³Defined in Equation (5.10).

hidden dimension of the LSTM layer and the directionality of the graph. For the Fully Connected Component, the only tunable hyperparameter is the activation function $\phi(\cdot)$, as the architecture is fixed to four layers with 128, 64, 32, and 16 neurons, respectively. Hyperparameter optimization is conducted using the Optuna framework (Akiba et al., 2019). Model training employs the Adam optimizer (Kingma and Ba, 2014), with the Mean Squared Error (MSE) augmented by an ℓ_2 regularization term as the loss function. The maximum number of training epochs is set to 1,000, and early stopping with a patience of 20 epochs is applied to mitigate overfitting. An incremental optimization strategy is adopted. First, the hyperparameters of the baseline models are optimized. Subsequently, for the Time-Geometric model, the optimized hyperparameters of the baseline model are retained for the Time Component, while only the hyperparameters of the Geometric Component are further optimized. Additional details regarding the hyperparameter configurations and their optimized values are provided in Appendix B.2.

5.4.3 Numerical Results

Forecasting Results

In this study, we focus on forecasting the *Close* price, denoted by S^* , and perform predictions at multiple temporal horizons q , namely daily, weekly, and monthly, corresponding to $q = 1, 5$, and 20 trading days, respectively. The feature matrix is constructed using the variables *Close*, *High*, *Low*, *Open*, and *Volume* for each stock under consideration.

Figure 19-21 and Table 29-31⁴ report the comparative results obtained for the different evaluation metrics between the proposed Time-Geometric model and the corresponding baseline models. For all metrics considered, lower values indicate superior predictive performance. To facilitate comparison across models, we compute the average value of each evaluation metric over the entire dataset for each model. Figure 19 summarizes the results for all considered architectures, both with the inclusion of a Graph Neural Network (i.e., the proposed Time-Geometric frame-

⁴Reported in Appendix B.3.

work) and without it (i.e., baseline models). The influence of geometric patterns becomes particularly pronounced when the Time-Geometric model employs Transformer and RCN architectures within the Time Component, where a clear improvement in average performance is observed. More generally, incorporating a GNN leads to a modest but consistent enhancement in the performance of several baseline models across multiple evaluation metrics. This behavior is consistent across longer forecasting horizons. Analogous trends are observed for predictions at 5 and 20 days, as illustrated in Figures 20 and 21, and in Tables 30 and 31.

Statistical Analysis

The adoption of multiple pairwise statistical tests⁵ is motivated by the heterogeneous nature of forecasting performance metrics and by the absence of a single test that is universally robust across all data conditions. Financial time series forecasting outcomes frequently exhibit non-normality, the presence of outliers, and heteroskedastic behavior. Consequently, relying on a single statistical test may yield biased or misleading conclusions. The combined use of parametric and non-parametric tests therefore enables a more robust and comprehensive assessment of performance differences between baseline models and their Time-Geometric extensions.

In this study, we compare $K = 16$ algorithms across $N_D = 90$ datasets. Table 4 reports the results of the pairwise statistical tests for the evaluation metrics considered in the one-day forecasting task. Entries labeled with *A* indicate acceptance of the null hypothesis, whereas those labeled with *R* denote its rejection; the corresponding test statistics are reported in parentheses. Moreover, the notation Δ -*Model* refers to the comparison between each baseline model and its counterpart augmented with GCNs. Results for the five-day and twenty-day forecasting horizons are provided in Tables 32 and 33, respectively, in Appendix B.3.

An inspection of Table 4 reveals that statistical significance is strongly dependent on the choice of evaluation metric. In particular, only Δ -

⁵Defined in Section 5.2.2.

Table 4: The table presents the outcomes of the tests, with Acceptance denoted as *A* and Rejection as *R*. The test statistics for the Paired t-test, Wilcoxon test, and Sign test are encapsulated in brackets. The entry Δ -“Model” signifies the comparison between the baseline model and its corresponding Time-Geometric model. For the Paired t-test, we consider 89 degrees of freedom, resulting in a critical value of 1.986. In the Wilcoxon test, the critical value is 1.9599, and for the Sign test, the critical value is 54.42.

MODEL	PAIRED T-TEST				WILCOXON TEST				SIGN TEST			
	RMSE	MAE	MAPE	MASE	RMSE	MAE	MAPE	MASE	RMSE	MAE	MAPE	MASE
Δ LSTM	A(0.4)	R(4.1)	R(4.6)	R(5.1)	A(1.2)	R(4.4)	R(4.5)	R(4.2)	A(50)	R(64)	R(64)	R(65)
Δ BiLSTM	A(0.4)	R(4.0)	R(4.9)	R(6.7)	A(1.6)	R(5.9)	R(5.9)	R(5.6)	A(58)	R(73)	R(73)	R(73)
Δ RNN	A(0.8)	A(0.5)	A(.03)	A(0.1)	A(0.2)	A(.03)	A(.04)	A(.05)	A(46)	A(47)	A(47)	A(47)
Δ BiRNN	A(1.0)	A(1.0)	A(1.9)	R(2.6)	A(0.7)	A(1.9)	R(2.3)	R(2.0)	A(40)	A(53)	A(53)	A(50)
Δ GRU	A(1.7)	R(6.1)	R(8.9)	R(8.8)	A(1.3)	R(7.5)	R(7.5)	R(7.3)	A(47)	R(82)	R(82)	R(80)
Δ BiGRU	R(2.7)	A(.01)	A(0.7)	A(0.6)	R(3.9)	R(2.3)	A(1.9)	A(1.3)	R(60)	A(54)	A(54)	A(53)
Δ TRANSF.	R(4.8)	R(7.0)	R(12)	R(17)	R(8.2)	R(8.2)	R(8.2)	R(8.2)	R(90)	R(90)	R(90)	R(90)
Δ TCN	R(7.8)	R(8.2)	R(13)	R(20)	R(8.2)	R(8.2)	R(8.2)	R(8.2)	R(90)	R(90)	R(90)	R(90)

Transformer, Δ -TCN, and Δ -RNN yield consistent statistical outcomes across all metrics. This observation suggests that the statistical relevance of incorporating geometric patterns, as well as the comparative performance of the models more generally, is influenced by the specific evaluation metrics employed. Similar patterns are observed for the five-day and twenty-day forecasting horizons, as reported in Tables 32 and 33 in Appendix B.3. Notably, for the five-day forecasting task, the results exhibit a more uniform trend, indicating that the prediction horizon itself plays a crucial role in determining the statistical significance of performance differences among models.

Relying exclusively on pairwise comparisons between each algorithm and its Time-Geometric extension may increase the likelihood of spuriously rejecting the null hypothesis. To address this issue, we additionally perform a multiple-comparison analysis aimed at controlling the probability of committing at least one Type I error across all comparisons, commonly referred to as the family-wise error rate (Demšar, 2006).

In the multiple-comparison setting, all algorithms are analyzed jointly for each dataset, considering one evaluation metric at a time. Specifically,

Table 5: The table displays the Friedman test results for each metric considered, with Acceptance denoted as A and Rejection as R , while the test statistics are reported in brackets. We consider 15 degrees of freedom, resulting in a critical value of 27.48.

METRIC	1-DAY	5-DAY	20-DAY
RMSE	R(1041)	R(1068)	R(654)
MAE	R(1025)	R(1034)	R(652)
MAPE	R(1025)	R(1034)	R(652)
MASE	R(1025)	R(1040)	R(653)

we employ the Friedman test⁶, which ranks the performance values v_u^j of each algorithm on each dataset. The results of the Friedman test are reported in Table 5. As shown in this table, the null hypothesis of the Friedman test is rejected for all evaluation metrics and for all forecasting horizons considered, indicating the existence of statistically significant performance differences among the algorithms.

To further investigate these differences, we apply the Nemenyi post-hoc test⁷, which enables pairwise comparisons among all algorithms. Two algorithms are considered to differ significantly in performance if the difference between their average ranks exceeds a critical difference. For the experimental setting adopted in this study, the critical difference is $CD = 2.50$.

Figures 22 - 24⁸ illustrate the results of the Nemenyi test for all evaluation metrics under one-day, five-day, and twenty-day forecasting horizons, respectively. In these figures, non-statistically significant differences are highlighted in yellow. An analysis of the ranking patterns across models and metrics leads to conclusions consistent with those drawn from the pairwise tests: both the choice of evaluation metric and the forecasting horizon substantially affect the assessment of statistical significance. Importantly, the results of the Time–Geometric model consistently highlight

⁶Defined in Section 5.2.2.

⁷Defined in Section 5.2.2.

⁸Reported in Appendix B.3.

the statistical relevance of incorporating geometric patterns to enhance the predictive performance of baseline time series forecasting models.

5.5 Conclusion

In this chapter, we introduce a novel ensemble model designed to jointly exploit temporal and geometric patterns in time series data through the integration of classical deep learning architectures with Graph Neural Networks (GNNs). In addition, we conduct an extensive statistical significance analysis to rigorously assess the contribution of geometric patterns within the forecasting framework. The chapter begins with a detailed description of the Time–Geometric model, including its overall architecture and constituent components. We then present the statistical methodology adopted to evaluate the relevance of geometric information, covering both pairwise and multiple-comparison testing procedures.

From an empirical perspective, we first demonstrate the performance gains achieved by the Time–Geometric model in one-step-ahead forecasting of the Standard and Poor’s 500 index when compared with several benchmark models. We subsequently perform a comprehensive statistical analysis. Specifically, we conduct pairwise comparisons using the paired t -test, the Wilcoxon signed-rank test, and the Sign test. The results indicate that the statistical significance of performance improvements strongly depends on the choice of evaluation metric. We then extend the analysis to a multiple-comparison setting by applying the Friedman test followed by the Nemenyi post-hoc test. These results further show that both the evaluation metric and the forecasting horizon play a critical role in determining statistical significance. Overall, the empirical and statistical evidence consistently demonstrates that the Time–Geometric model effectively captures and exploits geometric patterns, leading to statistically significant improvements in predictive performance over baseline time series forecasting models.

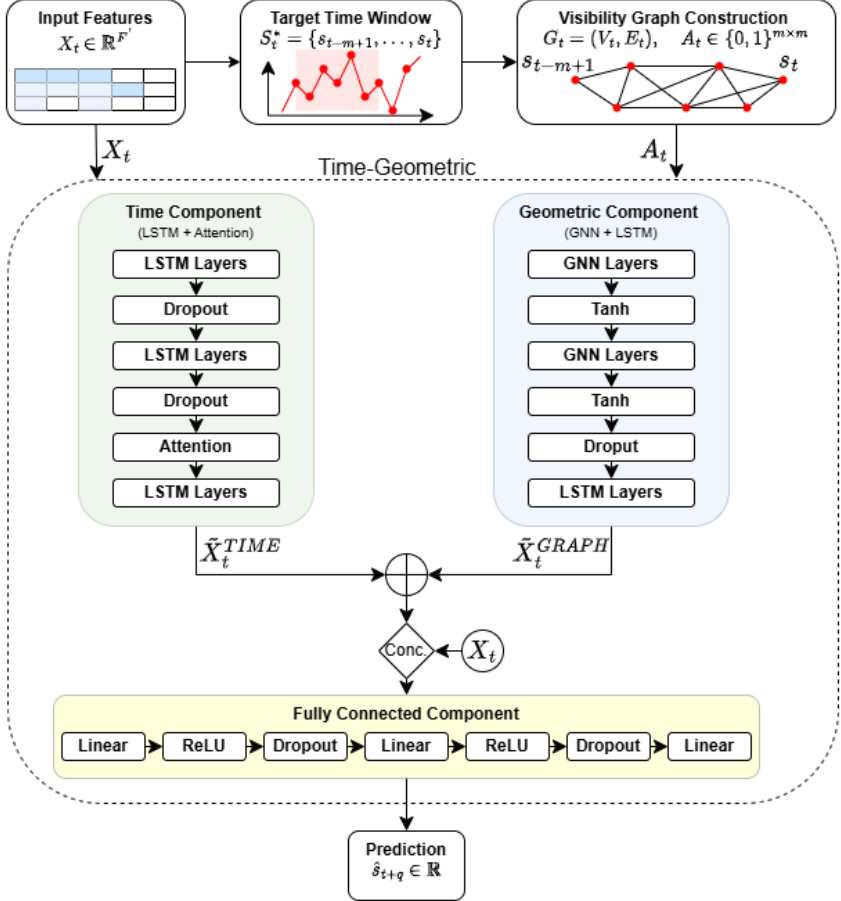


Figure 11: Time-Geometric model used to forecast the S&P 500.

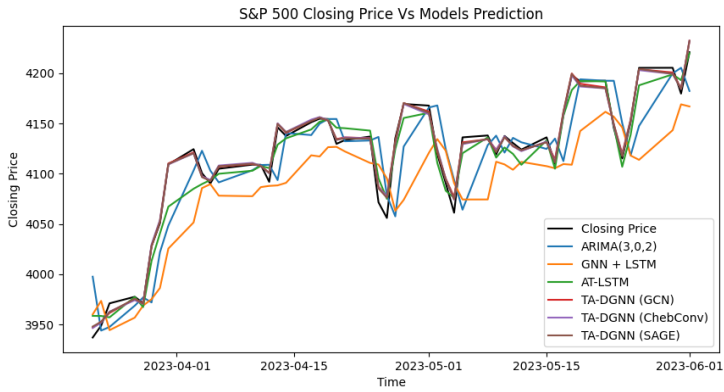


Figure 12: Comparisons among the models' predictions and the S&P 500 closing price from 22/03/2023 to 01/06/2023.

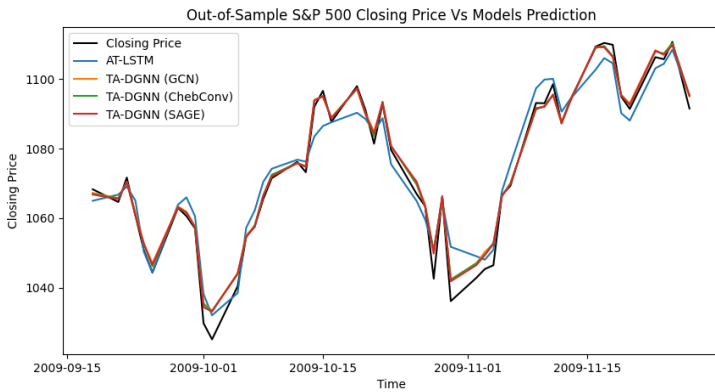


Figure 13: Comparisons among the models' predictions and the S&P 500 closing price from 19/09/2009 to 27/11/2009.

Chapter 6

Advances in Geometric Deep Learning

*"The board is set, the pieces are moving.
We come to it at last, the great battle of our time"*

Gandalf,
The Lord of The Rings: The Return of the King

Traditional graph representations model relationships exclusively through pairwise interactions. However, financial markets often exhibit collective dynamics that involves groups of assets driven by common economic factors, sectoral behavior, or macroeconomic events. Modeling these interactions using only pairwise edges may, therefore, fail to capture the higher-order dependencies inherent in financial systems. Moreover, financial time series exhibit complex temporal and geometric characteristics that pose significant challenges not only for forecasting but also for realistic data generation and relational learning. To address these challenges, this chapter investigates the use of Geometric Deep Learning for both generative modeling and higher-order representation learning. In particular, Graph Neural Networks and Hypergraph Neural Networks are employed to exploit relational structures beyond conventional sequential representations. In addition, Generative Adversarial Networks (GANs) are integrated with the Signature Transform to generate syn-

thetic financial time series that preserve both the statistical properties and the geometric characteristics of real market data. Using signatures as an additional learning objective, the proposed framework encourages the generator to reproduce higher-order temporal dependencies that are difficult to capture using traditional adversarial training alone.

Building upon this motivation, this chapter presents three research contributions. First, we propose a signature-guided graph-based GAN capable of generating realistic financial time series that reproduce the empirical return distributions of three stock exchanges (GREGNANIN, De Smedt, et al., 2025a; Marco Gregnanin et al., 2026a). Second, we extend this framework by demonstrating the effectiveness of geometric representations in generating synthetic financial data. Finally, we propose a novel Hypergraph Neural Network capable of inferring hypergraph structures directly from multivariate time series without requiring predefined relationships among variables (Marco Gregnanin et al., 2026b). This framework enables the discovery of higher-order interactions and latent group structures directly from the data. This chapter is based on one published work, Gregnanin, De Smedt, et al. (2025a), and two ongoing research projects (Marco Gregnanin et al., 2026a; Marco Gregnanin et al., 2026b). Together, these contributions demonstrate that Geometric Deep Learning provides a unified framework for addressing multiple financial time series tasks, including synthetic data generation and higher-order relational modeling.

6.1 Introduction

Across a wide range of application domains - including healthcare, signal and image processing, economics and finance, environmental sciences, and energy systems - the generation of synthetic data has become increasingly important. This demand is driven by multiple factors, such as limited data availability due to privacy constraints, the high cost of data collection, and the need to enrich datasets in order to improve the robustness and generalization capabilities of data-driven models. Despite its relevance, the generation of synthetic time series data remains a

challenging problem, primarily due to the intrinsic stochasticity, temporal dependence, and non-stationarity that characterize many real-world time series, and financial data in particular.

Two main paradigms are typically adopted for synthetic data generation: model-based and data-driven approaches. Model-based methods rely on explicitly specifying the underlying data-generating process through statistical or mathematical models. Classical statistical models, such as Autoregressive Integrated Moving Average (ARIMA) models (Box et al., 2015), assume stationarity or weak stationarity of the time series, implying time-invariant moments and autocovariance structures (Tsay, 2005). While these approaches have yielded useful insights and tractable models, their assumptions often fail to capture key empirical properties of financial time series, such as volatility clustering, regime shifts, and long-range dependencies. In contrast, data-driven approaches aim to learn latent structures directly from the data with fewer restrictive assumptions. In this context, machine learning and deep learning models - most notably Generative Adversarial Networks (GANs) (Goodfellow, Pouget-Abadie, et al., 2020) - have demonstrated promising results in synthetic data generation. However, most existing GAN-based approaches for time series focus primarily on autoregressive temporal dependencies. For financial time series, this focus may be insufficient, as it does not adequately capture non-stationary behavior and time-varying volatility, which play a central role in risk assessment and trading strategies (Buff, 2002). Volatility, in particular, is a fundamental quantity in risk management and portfolio allocation (Meucci, 2005).

In this chapter, we show that geometric patterns provide an essential and complementary perspective for the generation and analysis of synthetic time series data. By transforming time series from a Euclidean representation into a non-Euclidean, graph-based space, it becomes possible to uncover structural properties that are otherwise difficult to detect. Importantly, graph-based representations relax the need for strict stationarity assumptions and enable the analysis of evolving structural dependencies within the data. Graph Neural Networks (GNNs) (Scarselli et al., 2008) provide a natural computational framework for modeling

such geometric representations. In addition, we leverage the time series signature (Terry J Lyons, 1998; Terry Lyons, Hao Ni, and Oberhauser, 2014), which offers a universal and unique representation of sequential data. Building on these ideas, we propose a novel generative adversarial framework for synthetic financial time series generation that jointly exploits temporal dependencies and geometric structure. This is achieved by integrating two complementary tools - the visibility graph and the time series signature - into a unified GAN-based architecture.

Beyond synthetic data generation, this chapter also addresses a closely related and equally challenging problem: the construction of hypergraph representations from multivariate time series without relying on prior domain knowledge. Hypergraphs generalize standard graphs by allowing hyperedges to connect more than two nodes, thereby enabling the modeling of higher-order relationships (Bretto, 2013). Such relationships often encode critical information about collective dynamics and group-level interactions. While hypergraphs have been successfully applied in domains such as citation networks (Y. Feng et al., 2019), recommendation systems (L. Li and T. Li, 2013), social networks (Gao, Y. Feng, et al., 2022), and traffic forecasting (Yi and Park, 2020), their construction typically depends on predefined metadata or domain-specific assumptions. Deriving hypergraph structures directly from multivariate time series data, without prior knowledge, poses significant challenges. This task requires identifying latent groupings and higher-order dependencies among time series, a problem closely related to clustering under temporal dependence (Shun Wang et al., 2024). Standard clustering techniques, such as K-Nearest Neighbors (Peterson, 2009), are often inadequate in this setting, as they rely on static similarity measures and are sensitive to noise and scale variations. To address this limitation, we investigate data-driven methodologies for constructing dynamic hypergraph representations from multivariate time series. Our objective is to infer hypergraphs that capture non-pairwise dependencies and latent structural information using only the observed data. We propose two complementary approaches grounded in complex network theory (Barabási and

Pósfai, 2016). The first approach filters the time series correlation matrix using Random Matrix Theory (RMT) (Mantegna and Stanley, 1999) to retain statistically significant interactions, followed by modularity-based community detection (Fortunato, 2010) to identify clusters that define hyperedges. The second approach directly extracts inter-series dependencies through a self-attention mechanism (Cao et al., 2020), which allows to construct a graph from attention weights, and subsequently detect communities to form hypergraph structures. The resulting hypergraphs preserve the intrinsic network structure of the multivariate time series and are learned in a fully data-driven and task-adaptive manner. Finally, we exploit these higher-order representations by integrating Hypergraph Attention Networks (HANs) (S. Bai, F. Zhang, and Torr, 2021) with Long Short-Term Memory (LSTM) models (Hochreiter and Schmidhuber, 1997), augmented by attention mechanisms (Vaswani et al., 2017), to forecast future time series values across multiple datasets.

The remainder of this chapter is organized as follows. Section 6.2 presents a concise description of the proposed GAN model and illustrates the main results obtained. Section 6.3 illustrates the proposed hypergraph learning process and reports the main finding. Finally, Section 6.4 concludes the chapter.

6.2 Generative Adversarial Networks

In this section, we present the proposed Generative Adversarial Networks (GANs) framework. We begin by introducing the generative model and providing a detailed description of its architecture. Subsequently, we report a comprehensive numerical comparison between the proposed model and a set of baseline approaches in the task of generating synthetic financial time series data.

6.2.1 Proposed Framework

GANs constitute a class of data-driven machine learning models designed to learn complex probability distributions directly from data. A GAN is

formulated as a two-player adversarial game between two neural networks with opposing objectives: a generator and a discriminator. The generator seeks to produce synthetic data samples that resemble those drawn from an unknown target distribution, while the discriminator aims to distinguish between real data samples and those generated by the generator.

In this study, the generator, denoted by Gen , takes as input a random noise vector $\mathbf{z}$ and an adjacency matrix A , and, through a set of learnable parameters θ , produces a synthetic sample $\tilde{\mathbf{x}}$, namely

$$\tilde{\mathbf{x}} = Gen(\mathbf{z}, \theta, A).$$

Here, $Gen(\cdot)$ denotes the generator function, A is the adjacency matrix associated with the graph representation of the true data, and $\mathbf{z}$ is sampled from a predefined noise distribution $\mathbb{P}_{\mathbf{z}}(\mathbf{z})$, typically chosen as a multivariate normal or uniform distribution. Conversely, the discriminator, denoted by Dis , is tasked with distinguishing real data from synthetic data. It takes as input either a real sample $\mathbf{x}$ or a generated sample $\tilde{\mathbf{x}}$, together with the adjacency matrix A , and outputs the probability that the input sample is drawn from the true data distribution. Formally, with learnable parameters η , the discriminator is defined as

$$\mathbb{P}_{\text{real}}(\mathbf{x}) = Dis(\mathbf{x}, \eta, A),$$

where $Dis(\cdot)$ denotes the discriminator function. The generator and discriminator are trained simultaneously by minimizing and maximizing, respectively, a common value function, thereby constituting a zero-sum game. The training objective of the GAN is expressed as the following minimax problem:

$$\begin{aligned} \min_{Gen} \max_{Dis} V(Gen, Dis) &= \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_{\text{data}}(\mathbf{x})} [\log Dis(\mathbf{x}, A)] \\ &+ \mathbb{E}_{\mathbf{z} \sim \mathbb{P}_{\mathbf{z}}(\mathbf{z})} [\log(1 - Dis(Gen(\mathbf{z}, A), A))]. \end{aligned} \quad (6.1)$$

This adversarial training procedure results in the iterative improvement of the generator's ability to produce realistic synthetic samples and the discriminator's capability to identify generated data. In practice, the parameters of the discriminator and the generator, denoted by η and θ ,

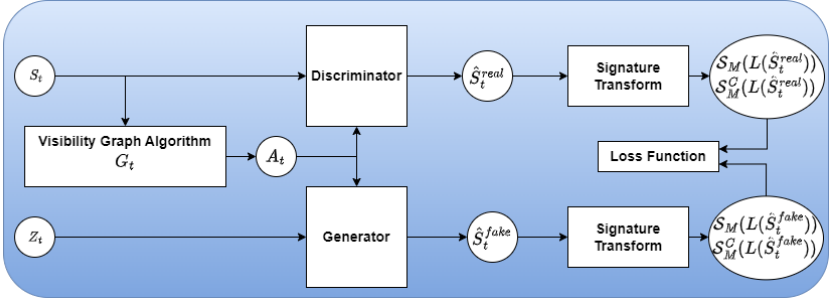


Figure 14: Proposed Generative Adversarial Networks Architecture.

respectively, are optimized by alternating gradient-based updates of the two networks.

Figure 14 illustrates the architecture of the GAN framework used. In line with the QuantGAN paradigm (Wiese et al., 2020), we adopt an identical architectural structure for both the generator and the discriminator. Let's consider a univariate time series $S = \{s_1, \dots, s_T\}$ with T observations. The procedure begins with the construction of a time series window $S_t = \{s_{t-m-1}, \dots, s_t\}$ with length m , and a corresponding noise matrix $Z_t \in \mathbb{R}^{m \times F'}$, where F' represents the number of sampling times. A graph representation G_t of the time series is then derived using the visibility graph algorithm, where the choice between a directed or undirected graph is treated as a hyperparameter. The resulting adjacency matrix $A_t \in \mathbb{R}^{m \times m}$ encodes the graph structure, with each node corresponding to a specific temporal observation. The discriminator and generator functions are thus defined as

$$Dis(S_t, \eta, A_t), \quad (6.2)$$

$$Gen(Z_t, \theta, A_t). \quad (6.3)$$

where $S_t \in \mathbb{R}^{m \times 1}$. In this work, both the generator and the discriminator are instantiated using the *Time-Geometric* model¹. For notational simplicity, the inputs to the generator and discriminator are denoted by Z_t and

¹A detailed description of the Time-Geometric model is provided in Chapter 5.2.

S_t , respectively, and are generically represented as $X_t \in \mathbb{R}^{m \times F'}$, where $F' = 1$ in the case of the discriminator.

6.2.2 Numerical Comparisons

Dataset and Pre-Processing

For the purposes of this study, we consider three main equity indices: Standard & Poor’s 500 (S&P500), Nasdaq Composite Index (IXIC), and Nikkei 225 (N225). For each index, we collect daily closing prices over the period Monday 4th January, 2010 to Monday 30th December, 2019 yielding approximately 2,515 observations per dataset. As a preprocessing step, we compute logarithmic returns $r_t = \log(s_t) - \log(s_{t-1})$, and subsequently standardize the resulting series to have zero mean and unit variance. To induce heavy-tailed behavior consistent with empirical return distributions, we apply the inverse Lambert- W transform (Goerg, 2015) following the procedure in Wiese et al. (2020). Empirical stock returns exhibit well-known stylized facts (Cont, 2001), including fat-tailed marginal distributions, volatility clustering (i.e., alternating regimes of high and low variability), and leverage effects, whereby negative returns are associated with subsequent increases in volatility.

Evaluation Metrics and Loss Functions

To evaluate the fidelity of the generated series, we adopt complementary metrics capturing both distributional and dependence properties. Concretely, we employ the leverage-effect score proposed in Wiese et al. (2020), the Earth Mover’s Distance (EMD), also known as the Wasserstein-1 distance (Rubner, Tomasi, and Guibas, 2000; Villani et al., 2009), to quantify discrepancies between empirical and generated distributions, and the Root Mean Square Error (RMSE) between the signatures of real and generated series.

As shown in Figure 14, before evaluating the loss of the model, we transform the output of the discriminator and generator, denoted $\hat{S}_t^*$ (with $* \in \{real, fake\}$), using the lead-lag map $L(\cdot)$. We then compute the trun-

cated signature² $\mathcal{S}_M(\cdot)$ with truncation degree $M = 5$. In addition, we compute the cumulative truncated signature $\mathcal{S}_M^C(\cdot)$ (Chevyrev and A. Kormilitzin, 2016) after applying a cumulative sum to $\hat{S}_t^*$, thereby retaining path-dependent information and reflecting the temporal accumulation of effects. Signatures are computed using the `iisignature` Python package (Kidger and Terry Lyons, 2020).

We consider two custom losses, Mean Squared Error (MSE) and Kullback–Leibler Divergence (KLD). Denoting by $\mathcal{S}_M(\cdot)$ and $\mathcal{S}_M^C(\cdot)$ the truncated and cumulative truncated signatures, respectively, we define:

$$\begin{aligned} \text{MSE}(\hat{S}_t^{fake}, \hat{S}_t^{real}) &= \frac{1}{N} \sum_{i=1}^N \left(\mathcal{S}_M(L(\hat{S}_t^{fake}))_i - \mathcal{S}_M(L(\hat{S}_t^{real}))_i \right)^2 \\ &\quad + \frac{1}{N} \sum_{i=1}^N \left(\mathcal{S}_M^C(L(\hat{S}_t^{fake}))_i - \mathcal{S}_M^C(L(\hat{S}_t^{real}))_i \right)^2, \\ \text{KLD}(\hat{S}_t^{fake}, \hat{S}_t^{real}) &= D_{KL} \left(\mathcal{S}_M(L(\hat{S}_t^{fake})) \| \mathcal{S}_M(L(\hat{S}_t^{real})) \right) \\ &\quad + D_{KL} \left(\mathcal{S}_M^C(L(\hat{S}_t^{fake})) \| \mathcal{S}_M^C(L(\hat{S}_t^{real})) \right), \end{aligned} \quad (6.4)$$

where N denotes the dimensionality of $\mathcal{S}_M(\cdot)$ (and $\mathcal{S}_M^C(\cdot)$), and $DKL(\cdot|\cdot)$ is the Kullback–Leibler divergence (Kullback and Leibler, 1951). To evaluate DKL , we apply a softmax transformation to $\mathcal{S}_M(\cdot)$ and $\mathcal{S}_M^C(\cdot)$ to obtain valid probability vectors. All experiments are run separately for each loss.

Hyperparameters Settings and Baseline Models

For both the generator and the discriminator, we adopt the *Time-Geometric* model. The temporal component consists of a two-layer LSTM network with a dropout layer applied between the two layers. The geometric component is composed of a two-layer Graph Convolutional Network (GCN) followed by an LSTM layer. The outputs of the temporal and geometric components are subsequently combined and fed into a Multilayer Perceptron (MLP) to produce the final prediction. We adopt an

²A detailed description of the Signature transform can be found in Section 3.3.

Table 6: Definition of the hyperparameters.

Loss	BS	LR	# Neurons	# Layers	Dropout	Seq. Len.	Dir
MSE	30	0.000797	[190, 120, 190]	[3, 7]	0.31	100	None
KLD	30	0.000221	[110, 70, 190]	[2, 4]	0.35	100	None

LSTM as the Temporal component to emphasize the incremental contribution of geometric structure: by combining a comparatively simple recurrent architecture with a GNN-model, we aim to demonstrate that the integration of geometric information can yield competitive - or superior - generation quality even relative to baselines that employ more complex temporal architectures (e.g., the TCN-based QuantGAN of Wiese et al. (2020)). Model training requires selecting hyperparameters, including batch size (BS), learning rate (LR), sequence length (Seq. Len.) for graph construction, graph direction (Dir.) in the visibility-graph procedure, the number of neurons and layers in both the Temporal and Geometric component, dropout rate, and related architectural choices. We use RMSProp (Hinton, Srivastava, and Swersky, 2012) for optimization and fix the number of training epochs to 100. Because the objective depends on the selected loss (Equation (6.4)), we perform two separate hyperparameter searches - one per loss function. The resulting optimal configurations are reported in Table 6; the neuron counts correspond to the GNN and LSTM modules in the Geometric component and to the LSTM module in the Temporal component, while the layer counts correspond to the GNN and LSTM depth. In all experiments, we consider undirected visibility graphs. As baselines, we adopt the QuantGAN configuration from Wiese et al. (2020) and a Monte Carlo simulation of geometric Brownian motion, whose parameters are estimated by maximum likelihood (Hilpisch, 2015).

Numerical Results

Synthetic data are generated at multiple temporal horizons - daily, weekly, monthly, and long-term - corresponding to 1, 5, 20, and 100 days, respectively. Quantitative results for real and generated series are reported

in Table 7 (best values in bold). Overall, the proposed Sig-Graph GAN consistently improves upon baselines in terms of EMD and the leverage-effect score. The preferred loss depends on the dataset: KLD yields better results for N225 and IXIC, whereas MSE performs best for S&P500. QuantGAN remains competitive on IXIC for the one-day horizon, achieving strong EMD and leverage-effect scores. For the Sig-RMSE metric, the proposed approach and geometric Brownian motion exhibit comparable performance, reflecting the fact that the Monte Carlo baseline is calibrated solely through first- and second-order moments.

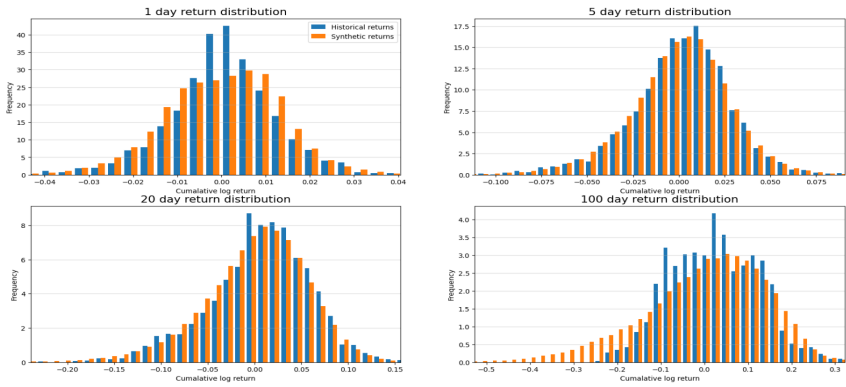


Figure 15: Sig-Graph GAN (KLD) Cumulative Returns Distribution of the N225 for the historical returns (blue) and for the synthetic returns (orange).

Figure 15 reports cumulative log-return distributions generated by the Sig-Graph GAN (KLD loss) on the N225 dataset, illustrating that the model reproduces the distributional shape particularly well for 5 and 10 days; for the one-day horizon, it captures tail behavior but underestimates mass in the central region.

Ablation Study To quantify the contribution of each architectural component, we perform an ablation study in which one element is removed at a time while keeping all other settings fixed. The components considered are the Geometric component, Temporal component, Feedforward Block, skip connection, and dropout. Figure 16 summarizes the results

Table 7: Results for the real and generated data for the Nasdaq (IXIC), Nikkei 225 (N225), and S&P 500 datasets. Values are multiplied by 100. For Sig-Graph GAN, the loss function is reported in parentheses. Best results are highlighted in bold.

(a) Nasdaq (IXIC)

Metric	QuantGAN	Sig-GAN (MSE)	Sig-GAN (KLD)	MC	GARCH
EMD(1)	0.1323	0.1649	0.1618	64.1325	79.6734
EMD(5)	0.4051	0.4949	0.3931	321.0774	175.3732
EMD(20)	1.4059	1.3431	1.3247	1284.3564	354.0737
EMD(100)	5.8784	5.4952	5.6074	6400.7781	799.9104
Sig-RMSE(1)	5.4381	5.4062	5.3985	5.3843	10490.6999
Sig-RMSE(5)	5.6048	5.5681	5.5703	5.5528	12185.4361
Sig-RMSE(20)	5.4281	5.3587	5.3450	5.4102	15932.3498
Sig-RMSE(100)	5.5419	5.4419	5.4224	5.3526	22124.6168
Leverage Effect	3.8110	4.8641	3.9694	4.1356	3.8905

(b) S&P 500

Metric	QuantGAN	Sig-GAN (MSE)	Sig-GAN (KLD)	MC	GARCH
EMD(1)	0.1483	0.1274	0.1472	50.3969	68.8657
EMD(5)	0.3681	0.3144	0.2816	252.3970	151.0271
EMD(20)	1.0954	0.8679	0.9438	1009.8142	305.4143
EMD(100)	4.2506	4.0865	4.3141	5034.7390	695.6773
Sig-RMSE(1)	4.1200	4.0911	4.0782	4.0809	5618.0861
Sig-RMSE(5)	4.2529	4.2228	4.2117	4.2123	5087.6136
Sig-RMSE(20)	3.9921	3.9763	3.9288	3.9120	6263.2784
Sig-RMSE(100)	4.6748	4.6810	4.6422	4.5633	6461.6048
Leverage Effect	3.8231	3.9510	3.6354	4.1250	3.9218

(c) Nikkei 225 (N225)

Metric	QuantGAN	Sig-GAN (MSE)	Sig-GAN (KLD)	MC	GARCH
EMD(1)	0.1759	0.1809	0.1611	40.1514	101.6870
EMD(5)	0.4341	0.4232	0.2066	203.1317	223.6870
EMD(20)	1.0522	1.4692	0.6767	815.9213	450.6941
EMD(100)	4.8196	4.1957	3.3477	4077.6804	1016.9772
Sig-RMSE(1)	8.1107	8.0068	8.0583	8.0334	23154.0991
Sig-RMSE(5)	7.7411	7.6333	7.6854	7.6637	25530.0890
Sig-RMSE(20)	7.7616	7.6707	7.7020	7.6722	24719.1718
Sig-RMSE(100)	8.4680	8.4262	8.4639	8.3349	54772.4263
Leverage Effect	3.5620	4.7748	3.0600	3.5627	3.4887

on N225 and S&P500 using EMD as the evaluation criterion, with ablated variants denoted, e.g., as “w/o Geometric Block”. The results indicate that removing the Geometric patterns consistently degrades performance under both loss functions, with a larger effect under KLD; by contrast, the influence of the dropout rate, skip connection, and feedforward block is comparatively stable across losses, and is most pronounced when training with MSE.

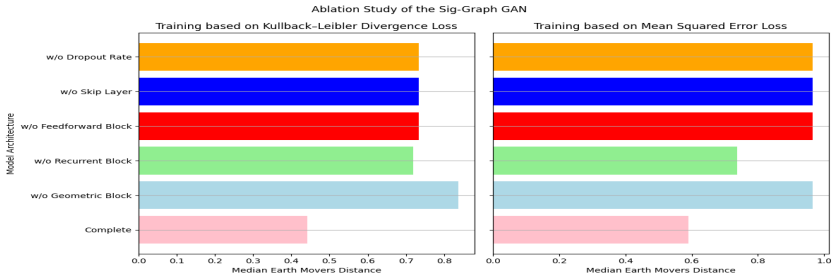


Figure 16: An ablation study was conducted for the Sig-Graph GAN trained using the two custom loss functions. The first plot refers to the Sig-Graph GAN (KLD) trained on the Nikkei 225 data, while the second plot refers to the Sig-Graph GAN (MSE) trained on the S&P 500.

6.3 Hypergraph Neural Networks

In this section, we present the proposed Dynamic Hypergraph Neural Networks (DHGNNs) framework. We begin by introducing the problem formulation, followed by a description of how hypergraph representations can be inferred from time series data without relying on prior knowledge. We then provide a detailed description of the DHGNN architecture. Finally, we present a comprehensive numerical comparison between the proposed model and a set of baseline approaches for the task of forecasting future values of time series.

6.3.1 Proposed Framework

Problem Formulation

Let $G^H = (\mathcal{V}^H, \mathcal{E}^H, \mathbf{W}^H)$ denote a hypergraph, where $\mathcal{V}^H = \{v_1, v_2, \dots, v_{N_V}\}$ is the set of nodes and $\mathcal{E}^H = \{e_1, e_2, \dots, e_{N_e}\}$ is the set of hyperedges. Each hyperedge $e_i \in \mathcal{E}^H$ is defined as a subset of the node set, i.e., $e_i \subseteq \mathcal{V}^H$. The diagonal matrix $\mathbf{W}^H \in \mathbb{R}^{N_e \times N_e}$ denotes the hyperedge weight matrix. In this work, we adopt a snapshot-based formulation to model dynamic hypergraphs (Antelmi et al., 2023). Under this paradigm, a dynamic hypergraph is represented as a sequence of static hypergraph snapshots observed at discrete time instants. Let $\mathcal{T} = \{t_1, t_2, \dots, t_{T_G}\}$ denote the set of timestamps, where $T_G = |\mathcal{T}|$. The dynamic hypergraph is then defined as

$$\mathcal{G}^H = \{G_{t_i}^H\}_{t_i=1}^{T_G},$$

where each snapshot $G_{t_i}^H = (\mathcal{V}_{t_i}^H, \mathcal{E}_{t_i}^H, \mathbf{W}_{t_i}^H)$ represents the hypergraph at time $t_i \in \mathcal{T}$. Notably, the node set, hyperedge set, and hyperedge weights are allowed to vary across snapshots. Let $\mathcal{S} = \{S_1, S_2, \dots, S_{N_s}\}$ denote a collection of N_s time series, where each s_i consists of T temporal observations. For simplicity, we assume that the number of time series, N_s , coincides with the number of nodes, N_V , in each hypergraph snapshot. Moreover, let $\mathcal{S}_t = \{S_1(t), S_2(t), \dots, S_{N_s}(t)\}$ denote the set of time series values observed at time t . We further assume that the sampling frequency of the time series and the timestamps in $\mathcal{T}$ coincide.

The objective of this study is to learn a function $h(\cdot)$ capable of predicting the future q values of a target time series $S^* \in \mathcal{S}$ by jointly exploiting the hypergraph representation and historical observations. Specifically, the model takes as input the hypergraph snapshot $G_{t_i}^H$ and the past m observations of each time series. Formally, the prediction task is defined as

$$\hat{S}_{t+q}^* = h(\mathbf{X}_t, G_{t_i}^H), \quad \forall t \in T, \quad \forall t_i \in \mathcal{T}, \quad (6.5)$$

where $\mathbf{X}_t \in \mathbb{R}^{m \times N_s}$ is the feature matrix collecting the past m observations of the N_s time series in tabular form, namely

$$\mathcal{S}_{t-m+1:t} = \{S_1(t-m+1:t), S_2(t-m+1:t), \dots, S_{N_s}(t-m+1:t)\}.$$

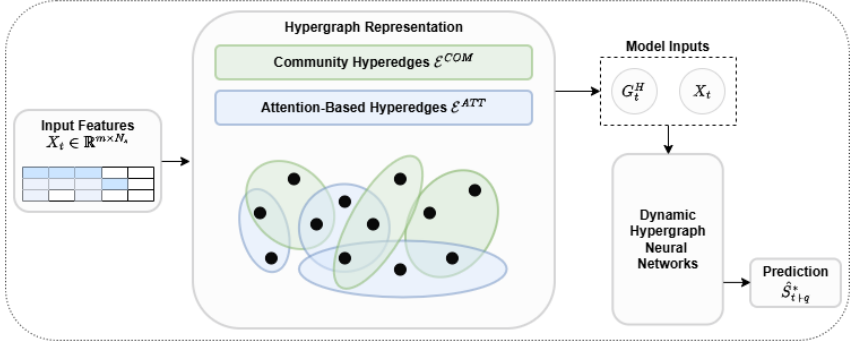


Figure 17: An overall pipeline of the proposed approach, showing all the steps needed to arrive at the final outcome starting from $\mathcal{S}$.

The hypergraph snapshot $G_{t_i}^H$ is constructed from these same historical observations, i.e., from $\mathcal{S}_{t-m+1:t}$. Consequently, the total number of hypergraph snapshots depends on the length of the time series T and the window size m , yielding $T_G = T - m + 1$.

An overview of the proposed framework is illustrated in Figure 17. The procedure consists of three main steps: extracting the past m observations from the multivariate time series; constructing the corresponding hypergraph representation $G_{t_i}^H$; and jointly leveraging $G_{t_i}^H$ and the feature matrix $\mathbf{X}_t$ to predict the future values $\hat{S}_{t+q}^*$.

Dynamic Hypergraph Representation

In our construction, hyperedges are obtained via community detection over two time-varying dependency estimates: a correlation matrix filtered through Random Matrix Theory (RMT), and a latent dependency matrix learned through a self-attention mechanism. Both mechanisms aim to identify groups of time series that exhibit stronger within-group dependence than cross-group dependence.

Community Hyperedges. Time series assigned to the same community are connected by a hyperedge $e_{t_i,j} \in \mathcal{E}_{t_i}^{\text{COM}}$, where

$$\mathcal{E}_{t_i}^{\text{COM}} = \{e_{t_i,1}, e_{t_i,2}, \dots, e_{t_i,N_{e_k}}\},$$

and N_{e_k} denotes the number of detected communities. This approach is motivated by the observation that communities in financial correlation networks often correspond to groups of series that are positively correlated within the group and comparatively negatively correlated across groups (MacMahon and Garlaschelli, 2015). At each time t , we compute an empirical correlation matrix $C_t \in \mathbb{R}^{N_V \times N_V}$ using the past m observations $\mathcal{S}_{t-m+1:t}$. Before performing community detection, we apply RMT-based filtering to remove noise and retain only statistically significant components³.

Attention-Based Hyperedges. The second mechanism constructs hyperedges from a latent dependency matrix inferred via self-attention. Starting from the input matrix $\mathbf{X}_t \in \mathbb{R}^{m \times N_s}$, we first apply a Gated Recurrent Unit (GRU) layer (Cho et al., 2014) to capture temporal dependencies. Following A. Zhang et al. (2023), the GRU update is:

$$\begin{aligned} \mathbf{R}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xr} + \mathbf{Z}_{t-1} \mathbf{W}_{zr} + \mathbf{b}_r), \\ \mathbf{U}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xu} + \mathbf{Z}_{t-1} \mathbf{W}_{zu} + \mathbf{b}_u), \\ \tilde{\mathbf{Z}}_t &= \tanh(\mathbf{X}_t \mathbf{W}_{xz} + (\mathbf{R}_t \odot \mathbf{Z}_{t-1}) \mathbf{W}_{zz} + \mathbf{b}_z), \\ \mathbf{Z}_t &= \mathbf{U}_t \odot \mathbf{Z}_{t-1} + (1 - \mathbf{U}_t) \odot \tilde{\mathbf{Z}}_t. \end{aligned} \tag{6.6}$$

Here $\mathbf{R}_t, \mathbf{U}_t \in \mathbb{R}^{m \times z_h}$ are the reset and update gates, $\tilde{\mathbf{Z}}_t \in \mathbb{R}^{m \times z_h}$ is the candidate hidden state, and $\mathbf{Z}_t \in \mathbb{R}^{m \times z_h}$ is the hidden state. The matrices $\mathbf{W}_{xr}, \mathbf{W}_{xu}, \mathbf{W}_{xz} \in \mathbb{R}^{N_V \times z_h}$ and $\mathbf{W}_{zr}, \mathbf{W}_{zu}, \mathbf{W}_{zz} \in \mathbb{R}^{z_h \times z_h}$ are learnable parameters, $\mathbf{b}_r, \mathbf{b}_u, \mathbf{b}_z \in \mathbb{R}^{1 \times z_h}$ are biases, $\odot$ denotes the Hadamard product, and z_h is the dimension of the hidden units.

We then define query and key matrices as $\mathbf{Q}_t = \mathbf{Z}_t \mathbf{W}_Q$ and $\mathbf{K}_t = \mathbf{Z}_t \mathbf{W}_K$,

³A detailed description of the procedure for performing community detection using Random Matrix Theory is provided in Chapter 4.2.

where $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{z_h \times N_s}$ are learnable. The resulting latent dependency matrix $\mathbf{A}_t \in \mathbb{R}^{N_s \times N_s}$ is computed by

$$\mathbf{A}_t = \text{softmax} \left(\frac{\mathbf{Q}_t^T \mathbf{K}_t}{\sqrt{N_V}} \right). \quad (6.7)$$

We interpret $\mathbf{A}_t$ as an attention-derived analogue of a correlation matrix and apply the same modularity-based community detection as in (3.2), replacing $\mathbf{C}_t^*$ with $\mathbf{A}_t$. The resulting communities produce attention-based hyperedges $e_{t_i, j} \in \mathcal{E}_{t_i}^{\text{ATT}}$, where

$$\mathcal{E}_{t_i}^{\text{ATT}} = \{e_{t_i, 1}, e_{t_i, 2}, \dots, e_{t_i, N_{e_k'}}\}$$

and $N_{e_k'}$ is the number of communities obtained from $\mathbf{A}_t$.

Hypergraph Assembly and Weighting We form the final hyperedge set by union:

$$\mathcal{E}_{t_i}^H = \mathcal{E}_{t_i}^{\text{COM}} \cup \mathcal{E}_{t_i}^{\text{ATT}}.$$

The hypergraph can be represented via an incidence matrix $\mathbf{H}_{t_i}^H \in \mathbb{R}^{|\mathcal{V}^H| \times |\mathcal{E}_{t_i}^H|}$ with entries

$$h_{t_i}(v, e_{t_i, j}) = \begin{cases} 1 & \text{if } v \in e_{t_i, j}, \\ 0 & \text{otherwise.} \end{cases}$$

Each hyperedge is assigned a positive weight $w_{t_i}(e_{t_i, j})$, stored in the diagonal matrix $\mathbf{W}_{t_i}^H \in \mathbb{R}^{|\mathcal{E}_{t_i}^H| \times |\mathcal{E}_{t_i}^H|}$. In particular, we define

$$w_{t_i}(e_{t_i, j}) = \sum_{v_h, v_k \in e_{t_i, j}, h \neq k} |a_t(v_h, v_k)|,$$

where $a_t(v_h, v_k)$ is the (h, k) -th entry of $\mathbf{A}_t$ in (6.7). Finally, the vertex degree $d_{t_i}(v)$ and the hyperedge degree $\delta_{t_i}(e_{t_i, j})$ are given by

$$\begin{aligned} d_{t_i}(v) &= \sum_{e_{t_i, j} \in \mathcal{E}_{t_i}^H} w_{t_i}(e_{t_i, j}) h_{t_i}(v, e_{t_i, j}), \\ \delta_{t_i}(e_{t_i, j}) &= \sum_{v \in \mathcal{V}_{t_i}^H} h_{t_i}(v, e_{t_i, j}). \end{aligned}$$

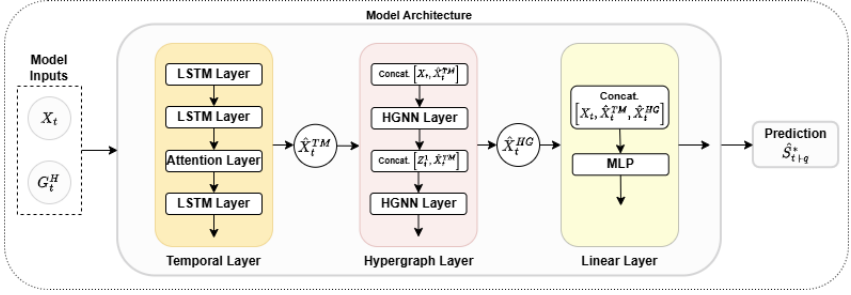


Figure 18: Proposed Hypergraph Neural Networks Architecture.

Let $\mathbf{D}_{t_i}^v \in \mathbb{R}^{|\mathcal{V}_{t_i}^H| \times |\mathcal{V}_{t_i}^H|}$ and $\mathbf{D}_{t_i}^e \in \mathbb{R}^{|\mathcal{E}_{t_i}^H| \times |\mathcal{E}_{t_i}^H|}$ denote the corresponding diagonal matrices of vertex degrees and hyperedge degrees at time $t_i \in \mathcal{T}$, respectively.

Several alternative hypergraph construction strategies exist, including K-nearest-neighbor hypergraphs (X. Li et al., 2014; Y.-T. Wang et al., 2021), sector-based hypergraphs (Wei et al., 2025), and correlation-based hypergraphs (Jian Huang et al., 2025). These methods either depend on predefined similarity measures or require prior domain knowledge regarding time series relationships. In contrast, the proposed framework aims to construct the hypergraph directly from the temporal and geometric characteristics of the data, thereby reducing the reliance on manually specified relational structures. Nevertheless, a systematic comparison with these alternative hypergraph construction strategies would provide valuable insight into the influence of graph construction on predictive performance and constitutes an important research for future work.

Dynamic Hypergraph Neural Networks

Figure 18 shows the model architecture of the proposed Dynamic Hypergraph Neural Networks (DHNN). The model takes as input the feature matrix $X_t \in \mathbb{R}^{m \times N_s}$ and the hypergraph representations G_t^H , in order to predict the future q values of the target time series S_{t+q}^* , as defined in equation 6.5. As shown in Figure 18, the proposed hypergraph neural network architecture is composed of three different layers; a *Temporal*

Layer, which captures the temporal dependencies; a *Hypergraph Layer*, which models the high-order relationship; and a *Linear Layer*, which integrates the learned representation to produce the final forecast.

Temporal Layer The temporal layer takes as input $X_t \in \mathbb{R}^{m \times N_s}$, and produces a temporal representation $\tilde{X}_t^{TM} \in \mathbb{R}^{m \times N_s}$. This module combines a stacked LSTM layer with a multi-head self-attention mechanism, followed by a fully connected transformation. Its main objective is to extract a compact latent representation of the input sequence while preserving both short-term and long-term temporal structures.

First, the input is processed through two stacked LSTM layers:

$$H_t^{(1)} = LSTM_1(X_t), \quad H_t^{(2)} = LSTM_2(H_t^{(1)}),$$

where $H_t^{(1)} \in \mathbb{R}^{m \times h_1}$, $H_t^{(2)} \in \mathbb{R}^{m \times d}$, with h_1 the hidden dimension of the first LSTM layer, $d = h_2 \cdot n_{head}$ is the hidden dimension of the second LSTM layer with h_2 representing the number of neurons and n_{head} the number of attention heads. A multi-head self-attention mechanism is then applied:

$$\tilde{H}_t = \text{MultiHeadAttention}(H_t^{(2)}),$$

where $\tilde{H}_t \in \mathbb{R}^{m \times d}$ which allows the model to assign different importance weights to different time steps. We can define the attention score using the scaled dot-product as:

$$A_t^{(k)} = \frac{Q_t^{(k)} (K_t^{(k)})^T}{\sqrt{d_k}}, \quad \forall k = 1, \dots, n_{head}$$

where $Q_t^{(k)} = H_t^{(2)} W_Q^{(k)}$ are the queries, $K_t^{(k)} = H_t^{(2)} W_K^{(k)}$ the keys, and $W_Q^{(k)}, W_K^{(k)} \in \mathbb{R}^{d \times d_k}$ are learnable projection matrices with $d_k = d/n_{head}$. A softmax function is applied row-wise to obtain attention weights:

$$\alpha_t^{(k)} = \text{softmax} \left(A_t^{(k)} \right).$$

Then, the output of the k -th head is:

$$\text{head}_t^{(k)} = \alpha_t^{(k)} V_t^{(k)},$$

where $V_t^{(k)} = H_t^{(2)} W_V^{(k)}$ are the values with $W_V^{(k)} \in \mathbb{R}^{d \times d_k}$ learnable matrix weights. Finally, the outputs of the heads are concatenated as:

$$\text{MultiHead} \left(H_t^{(2)} \right) = \parallel_k^{n_{head}} \text{head}_t^{(k)} \in \mathbb{R}^{n \times d}.$$

A final linear transformation is applied:

$$\tilde{H}_t = \text{MultiHead} \left(H_t^{(2)} \right) W_0,$$

where $W_0 \in \mathbb{R}^{d \times d}$ is a learnable projection matrix.

The resulting sequence is further processed by a third LSTM layer:

$$H_t^{(3)} = \text{LSTM}_1 \left(\tilde{H}_t \right).$$

Before obtaining the temporal representation $\tilde{X}_t^{TM}$, we apply a fully connected layer at $H_t^{(3)}$.

Hypergraph Layer The hypergraph layer takes as input $X_t \in \mathbb{R}^{m \times N_s}$, $\tilde{X}_t^{TM} \in \mathbb{R}^{m \times N_s}$, and the hypergraph representation G_t^H , and produces the output $\tilde{X}_t^{HG} \in \mathbb{R}^{m \times N_s}$. The purpose of this module is to capture higher-order structural dependencies among variables through a learnable hypergraph representation.

First, the temporal representation $\tilde{X}_t^{TM}$ is concatenated with the initial input X_t . Then, it is processed through a sequence of convolutional hypergraph layers. The first hypergraph convolution is defined as:

$$\begin{aligned} Z_t^{(0)} &= \left[X_t \parallel \tilde{X}_t^{TM} \right], \\ Z_t^{(1)} &= \sigma \left(\text{HGNN}_1 \left(Z_t^{(0)}, G_t^H \right) \right), \end{aligned}$$

where $\sigma(\cdot)$ is an Exponential Linear Unit (ELU)⁴ activation function (Clevert, Unterthiner, and Hochreiter, 2016). Then, the output is concatenated

⁴The ELU activation function is defined as:

$$\text{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

with $\alpha > 0$

with the temporal representation:

$$Z_t^{(\text{cat})} = \left[Z_t^{(1)} \| \tilde{X}_t^{TM} \right],$$

Then a linear transformation is applied:

$$\hat{Z}_t^{(1)} = W_1 Z_t^{(\text{cat})},$$

where W_1 is a learnable weight matrix. Finally, a second hypergraph convolutional layer refines the representation:

$$\tilde{X}_t^{\text{HG}} = \sigma \left(\text{HGNN}_2 \left(\hat{Z}_t^{(1)}, G_t^H \right) \right).$$

Note that a generic hypergraph convolutional layer, i.e. $\text{HGNN} (X_t, G_t^H)$, is defined as:

$$Z_t = \sigma \left(D_t^{v^{-1/2}} H_t^H W_t^H D_t^{e^{-1}} H_t^{H^T} D_t^{v^{-1/2}} X_t \Theta \right),$$

where X_t is the feature matrix at time t , Θ is the learnable weight matrix, D_t^v is the node degree matrix at time t , D_t^e is the edge degree matrix at time t , W_t^H is the diagonal matrix of hyperedge weights at time t , and H_t^H is the incidence matrix at time t .

Linear Layer The linear layer takes as input $X_t \in \mathbb{R}^{m \times N_s}$, $\tilde{X}_t^{TM} \in \mathbb{R}^{m \times N_s}$, and $\tilde{X}_t^{\text{HG}} \in \mathbb{R}^{m \times N_s}$ in order to predict the future q values of the target time series $S^* \in \mathcal{S}$.

First, we concatenate the three inputs:

$$\tilde{X}_t^{\text{cat}} = \left[\tilde{X}_t^{\text{TM}} \| \tilde{X}_t^{\text{HG}} \| X_t \right].$$

Then, the final prediction is obtained through a multi-layer perceptron (MLP):

$$\hat{S}_{t+q} = \text{MLP} \left(\tilde{X}_t^{\text{cat}} \right),$$

where $\hat{S}_t \in \mathbb{R}^q$ represents the predicted output over a forecast horizon q .

6.3.2 Numerical Comparisons

Dataset and Pre-Processing

For the purposes of our analysis, we consider three distinct multivariate time series datasets corresponding to different application domains: stock returns prediction, appliances' energy consumption forecasting (Candanedo, 2017), and air quality prediction (Vito, 2008).

For the first dataset, the objective is to predict the future value of the S&P500 index, i.e., the target time series, based on the constituent stocks within the index. We collect daily observations from Tuesday 13th July, 2010 to Friday 7th July, 2023, resulting in 3269 observations for each of the 500 index components, downloaded from Yahoo Finance⁵. After removing stocks with missing values and computing logarithmic returns, the dataset consists of 440 stocks, each with 3268 observations. To reduce the dimensionality and mitigate computational constraints in terms of memory usage and time complexity, we randomly select 10% of the available stocks to construct the final dataset. The second dataset concerns the prediction of appliances' energy consumption. It comprises temperature and humidity measurements collected every 10 minutes over a period of approximately 4.5 months within a residential environment, yielding 19735 observations. Measurements are obtained from multiple zones within a house, as well as from external environmental conditions, resulting in a total of 28 features. The third dataset focuses on air quality prediction in an Italian city. It consists of hourly averaged responses from an array of five metal oxide chemical sensors, collected between March 2004 and February 2005, for a total of 9558 observations. After removing categorical variables and imputing missing values using the most recent available observations, the dataset comprises 15 features.

To effectively capture nonlinear patterns, all input features across the three datasets are normalized prior to model training using a rolling window approach. Specifically, for each feature, the mean and standard deviation are computed over a sliding window of size w , and the normal-

⁵<https://finance.yahoo.com>

ized value is obtained as:

$$\tilde{x}_t^i = \frac{x_t^i - \mu_t^i}{\sigma_t^i},$$

where i denotes the feature index. This normalization procedure ensures that the data are appropriately scaled while preserving their temporal dynamics.

Baseline Models and Evaluation Metrics

For benchmarking purposes, we consider the following baseline models: a Graph Convolutional Network (GCN) combined with an LSTM layer, an attention-based LSTM (AT-LSTM), an attention-based Gated Recurrent Unit (AT-GRU), an attention-based bidirectional LSTM (AT-BiLSTM), a convolutional neural network augmented with an attention-based LSTM (CNN-AT-LSTM), a Transformer, and the *Time-Geometric* (TG) model. The selection of these baselines is motivated by the need to evaluate the performance gains achieved by incorporating hypergraph structures across different modeling paradigms. In particular, we compare purely temporal models (e.g., AT-LSTM, AT-GRU, Transformer and AT-BiLSTM), hybrid models that combine temporal and pairwise structural information (e.g., CNN-AT-LSTM and GCN with LSTM), and ensemble architectures (e.g., TG). This comparison aims to highlight the importance of capturing higher-order relationships among feature variables, as opposed to relying solely on pairwise dependencies typically modeled through graph-based approaches.

For graph-based models, the graph representation is constructed from the correlation matrix computed over the features in each dataset. Specifically, we apply the Planar Maximally Filtered Graph (PMFG) (Tuminello et al., 2005) to filter the correlation matrix, retaining only the most significant correlations while enforcing planarity, i.e., ensuring that the resulting graph can be embedded in a plane without edge crossings. In contrast, for the TG model, the graph representation is derived using the visibility graph algorithm applied to the target time series of each dataset.

Table 8: Definition of the hyperparameters.

Dataset	lr	α	# Layer	# tl	#h	w	head	BS	Dropout
Stock Returns	7.545×10^{-6}	0.3	1	50	15	200	6	32	0.1
Energy Consumption	5.815×10^{-6}	0.3	5	15	25	130	2	32	0.4
Air Quality	9.083×10^{-5}	0.3	5	35	35	100	4	32	0.2

To evaluate and compare the predictive performance of the considered models, we adopt several standard metrics for time series forecasting, namely Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). The formal definitions of these evaluation metrics are provided in Appendix B.1.

Hyperparameters Setting

To train the proposed framework, each dataset is partitioned into training, validation, and test sets using a 0.7-0.15-0.15 split, respectively. The validation set is employed to mitigate overfitting through the use of an early stopping strategy with a tolerance of 10 iterations. Model optimization is performed using the Adam optimizer (Kingma and Ba, 2014), with the number of training epochs fixed at 100. The Mean Squared Error is adopted as the loss function, and hyperparameter optimization is conducted using the Optuna Python package (Akiba et al., 2019).

For the proposed framework, the set of hyperparameters to be optimized includes: the learning rate (lr), the α parameter of the ELU activation function (α), the number of temporal layers (# layers), the number of neurons in the temporal layers (#tl), the number of neurons in the hypergraph neural network (#h), the time window size (w), the number of attention heads (head), the batch size (BS), and the dropout rate (Dropout). It is important to note that the number of layers in the hypergraph neural network is fixed to two, as increasing the depth leads to a significant degradation in performance (Jing Huang and J. Yang, 2021). The selected hyperparameter values for each dataset are reported in Table 8. The hyperparameters of the baseline models are selected following the same optimization strategy, and are reported in Tables 35-37 in Appendix C.1.

Table 9: Comparison of forecasting models across datasets and evaluation metrics.

Model	Stock Returns			Energy Consumption			Air Quality		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
AT-LSTM	0.01310	0.00982	106.52	90.72	53.67	60.47	1.30001	1.00799	96.69
AT-GRU	0.01309	0.00984	104.13	90.78	53.95	61.10	0.96535	0.70244	55.69
BiAT-LSTM	0.01309	0.00983	103.82	90.40	51.14	54.84	1.30002	1.00804	96.70
CNN-AT-LSTM	0.01309	0.00983	103.29	90.79	53.95	61.09	1.16101	0.87183	74.88
Transformer	0.01381	0.01028	255.29	89.98	50.63	54.66	0.69199	0.47588	37.14
GCN with LSTM	0.00991	0.00726	103.17	65.63	31.57	28.82	0.64949	0.45108	34.34
TG	0.01309	0.00983	108.52	90.39	50.88	54.24	1.29985	1.00610	96.18
HGNN	0.01316	0.00981	146.35	61.03	29.85	29.41	0.64901	0.45745	34.98

Numerical Results

Table 9 reports the numerical results obtained across all the considered datasets. The best value for each evaluation metric within each dataset is highlighted in bold. From the table, it can be observed that the HGNN model does not consistently outperform the baseline models; rather, its performance depends strongly on the characteristics of the dataset. In highly stochastic settings, such as the stock returns dataset, the proposed framework exhibits relatively weaker performance. A plausible explanation is that, in such contexts, higher-order relationships contribute less to the predictive task, while pairwise dependencies are more informative. This interpretation is supported by the superior performance of the GCN combined with an LSTM, which explicitly models pairwise interactions. Conversely, for datasets where higher-order relationships are more likely to be present - such as energy consumption and air quality - the proposed HGNN framework achieves competitive results. In the air quality dataset, the model attains the lowest values for two out of three evaluation metrics, while in the energy consumption dataset it achieves the best result for one metric and remains close to the best performance in the others. These findings suggest that hypergraph-based representations are particularly effective when complex, non-pairwise dependencies exist among the variables. It is also important to highlight the poor performance of the *Time-Geometric* model. This can be attributed to the construction of the graph representation via the visibility graph using

a feature selected at random from the available variables. This design choice was made to prevent potential data leakage and avoid introducing bias. In particular, constructing the graph directly from the target variable would provide the model with information that should not be available at prediction time, thereby compromising the validity of the evaluation.

6.4 Conclusion

In this chapter, we introduced two geometric deep learning frameworks: one aimed at generating synthetic financial time series data, and the other designed to model and exploit higher-order relationships within time series. We began by presenting a Generative Adversarial Network (GAN) architecture for synthetic data generation, complemented by two custom loss functions based on the Signature Transform, which enforce the preservation of higher-order statistical properties. Subsequently, we proposed a novel methodology for deriving hypergraph representations from multivariate time series without relying on prior structural knowledge. In conjunction with this representation, we introduced a Hypergraph Neural Network (HGNN) framework to effectively capture and leverage higher-order dependencies among variables.

From an empirical perspective, we first evaluated the generative model by replicating the return distributions of three major stock indices - Nasdaq, Standard & Poor's 500, and Nikkei 225 - demonstrating that the proposed approach outperforms baseline models in capturing the distributional characteristics of financial returns. We then assessed the performance of the HGNN model across three distinct time series datasets: stock returns, energy consumption, and air quality. The results indicate that the proposed framework achieves strong performance for datasets where higher-order relationships are expected, such as energy consumption and air quality. Conversely, its performance deteriorates in highly stochastic settings, such as stock returns, where higher-order dependencies appear to be less informative than the pairwise relationship.

Chapter 7

Conclusion

"Well, here at last, dear friends, on the shores of the Sea comes the end of our fellowship in Middle-earth. Go in peace! I will not say: do not weep; for not all tears are an evil."

Gandalf,

The Lord of The Rings: The Return of the King

The primary objective of this thesis was to demonstrate the importance of incorporating geometric patterns into financial time series analysis. In particular, this work combines geometric representations with classical temporal patterns to model, forecast, and generate financial time series. Motivated by the limitations of traditional econometric approaches and purely temporal neural architectures, the thesis set out to develop a unified methodological framework capable of capturing the complex structural, nonlinear, and distributional properties that characterize financial markets.

The first part of the thesis focused on the comparative analysis of multiple financial time series through the Signature Transform. Chapter 4 introduced a paradigm shift from correlation-based similarity measures to a signature-based similarity matrix. This shift is motivated by the inherent limitations of correlation matrices, which are restricted to linear dependencies and disregard temporal ordering. In contrast, the signature-based similarity matrix provides a richer and more expressive charac-

terization of temporal dynamics and nonlinear dependencies by exploiting the algebraic and geometric properties of path signatures. Empirical results on the Standard & Poor's 500 (S&P500) index showed that signature-based similarity matrices improved community detection outcomes relative to correlation-based methods, both in terms of higher modularity values and greater consistency in the number of detected communities. Moreover, the signature-based approach was shown to be economically meaningful and advantageous for portfolio allocation and risk management, outperforming traditional covariance- or correlation-based frameworks by increasing expected returns and reducing portfolio volatility under market-neutral strategies.

Building on these results, the second part of the thesis addressed the problem of financial time series forecasting. In Chapter 5, the *Time-Geometric* model was proposed, combining a temporal component based on established deep learning architectures (e.g., Long Short-Term Memory networks) with a geometric component based on Graph Neural Networks. The graph representation of the time series is constructed using the visibility graph algorithm, allowing structural information to be explicitly modeled. A key advantage of the proposed framework lies in its modularity, as both the temporal and geometric components can be tailored to the specific task. The *Time-Geometric* model was evaluated against several baseline approaches for predicting S&P500 features and demonstrated superior performance across all considered evaluation metrics. Importantly, Chapter 5 also showed that these improvements are not merely empirical but statistically significant, underscoring the relevance of geometric patterns in forecasting tasks and highlighting the sensitivity of statistical conclusions to the choice of evaluation metrics in financial applications.

The third part of the thesis extended geometric deep learning methodologies to generative modeling and higher-order dependency analysis. In Chapter 6, Generative Adversarial Networks integrating temporal architectures and graph-based components were developed to generate synthetic financial time series. A central contribution of this chapter is the introduction of a signature-based loss function, designed to enforce the

preservation of higher-order distributional characteristics in the generated data. The proposed generative models were shown to accurately reproduce the return distributions of multiple stock exchanges, capturing both temporal dependencies and complex statistical features. In the same chapter, a novel data-driven methodology for constructing hypergraph representations from multivariate time series without prior structural knowledge was presented. By combining Hypergraph Neural Networks with recurrent architectures, the proposed framework successfully modeled non-pairwise dependencies and improved predictive performance across diverse datasets.

Overall, this thesis presents a coherent and unified framework that integrates graph-based learning, signature analysis, and geometric deep learning for financial time series analysis, explicitly leveraging both temporal and geometric patterns. The main contributions of this research can be summarized as follows: (i) the introduction of a signature-based similarity matrix for comparing financial time series; (ii) the development of ensemble models that combine classical temporal deep learning architectures with Graph Neural Networks to jointly capture temporal and geometric patterns; (iii) the design of generative models that exploit Graph Neural Networks and the Signature Transform to replicate stock return distributions; and (iv) the proposal of a fully data-driven model capable of learning hypergraph representations from multivariate time series without prior knowledge.

Despite these contributions, several limitations remain. In particular, the computational complexity associated with graph and hypergraph construction, as well as with the corresponding deep learning models, may hinder scalability to very large datasets. Further research is therefore required to develop more efficient or approximate solutions. Second, while the proposed models improve predictive accuracy across multiple tasks, they also reduce model interpretability. The combination of multiple black-box components increases the need for explainability, especially at the level of individual predictions. Third, empirical evaluation is primarily conducted on large-cap U.S. equities, particularly the S&P500, over specific historical periods. Consequently, the generalizabil-

ity of the proposed framework to other financial markets, such as foreign exchange, cryptocurrencies, fixed-income securities, or emerging markets, remains an open research question. Fourth, although the proposed methods consistently outperform the considered baselines, the influence of several design choices - including the visibility graph construction, the signature truncation level, and the similarity threshold used for network generation - was not investigated through a systematic sensitivity analysis. Such analyses would provide further insight into the robustness of the proposed framework. Fifth, from a financial perspective, the portfolio allocation experiments do not explicitly incorporate practical considerations such as transaction costs, portfolio turnover, market impact, or comparisons with simple benchmark strategies such as momentum-based allocation. Consequently, the reported improvements should be interpreted primarily as evidence of improved representation learning rather than definitive evidence of superior investment performance. Finally, financial markets are inherently non-stationary and characterized by structural breaks, regime shifts, and changing volatility conditions. Although the proposed models were evaluated across multiple datasets and forecasting tasks, their robustness under different market regimes, including periods of financial crises or extreme market stress, remains to be systematically investigated. Addressing these limitations represents an important direction for future research and would further strengthen the practical applicability and generalizability of the proposed geometric deep learning framework.

Several promising directions for future research naturally emerge from this work. First, the proposed frameworks could be applied not only to forecasting asset prices but also to the analysis of financial market stability, which is of particular relevance to policymakers. Second, the methodologies developed in this thesis may be extended to other application domains, such as energy markets, climate modeling, and healthcare time series. Third, future work could focus on integrating the Signature Transform more deeply into geometric deep learning architectures, rather than using it solely as an input representation or a loss function. Finally, the development of scalable and online graph and hypergraph

construction methods represents an important research project for extending these approaches to real-time and large-scale time series environments.

In conclusion, this PhD thesis demonstrates that the integration of temporal, geometric, and algebraic representations provides a powerful and flexible paradigm for financial time series analysis. By bridging ideas from stochastic analysis, network theory, and deep learning, this work advances the state of the art in modeling, forecasting, and generating complex financial data, and lays a solid foundation for future research at the intersection of geometric deep learning and quantitative finance.

Appendix A

Signature-Based Similarity

A.1 Filtering Method

Table 10 reports the critical values of the standard normal distribution.

Table 10: Critical values of the standard normal distribution.

Confidence Level	0.95	0.98	0.99	0.999
Critical Value	1.96	2.326	2.575	3.291

A.2 Asset Allocation Models

Let $\mathbf{S}$ denote a collection of N price time series, each observed over T discrete time points. Let $\mathbf{R}$ denote the corresponding collection of log-return time series, where each R_i consists of $T - 1$ observations. Specifically, for a generic asset i , whose price series is given by $S_i = \{s_i(1), \dots, s_i(T)\}$, the log-returns are defined as $R_i(t) = \log(s_i(t)) - \log(s_i(t - 1))$ for all $t \in \{2, \dots, T\}$. Finally, let $\mu \in \mathbb{R}^N$ denote the vector of expected log-returns estimated from $\mathbf{R}$, and let $\Sigma \in \mathbb{R}^{N \times N}$ denote the corresponding covariance matrix.

Mean-Variance Portfolio Strategy. The Mean–Variance (MV) portfolio strategy, originally introduced by Markowitz (Markowitz, 1952), constitutes a foundational framework in portfolio theory. The objective is to identify a portfolio allocation that optimally balances expected return and risk, where expected return is quantified by the mean vector μ , and risk is measured through the covariance matrix Σ . Formally, the mean-variance optimization problem is given by:

$$\begin{aligned} \min_w \quad & w^T \Sigma w - w^T \mu, \\ \text{s.t.} \quad & \sum_{i=1}^N w_i = 1, \\ & 0 \leq w_i \leq 1, \forall i = 1, \dots, N. \end{aligned} \tag{A.1}$$

where $w = (w_1, \dots, w_N)^T$ denotes the portfolio weight vector. The first constraint enforces a full investment budget, while the second constraint excludes short-selling.

Global Minimum Variance Portfolio Strategy. In contrast to the MV framework, the Global Minimum Variance Portfolio (GMVP) strategy focuses exclusively on risk minimization, independently of expected returns. The objective is to determine the portfolio weights that minimize the overall portfolio variance. The corresponding optimization problem is formulated as:

$$\begin{aligned} \min_w \quad & w^T \Sigma w, \\ \text{s.t.} \quad & \sum_{i=1}^N w_i = 1, \\ & 0 \leq w_i \leq 1, \forall i = 1, \dots, N. \end{aligned} \tag{A.2}$$

Maximum Sharpe Ratio Portfolio Strategy. The Sharpe ratio, introduced by Sharpe (Sharpe, 1998), is a widely used performance metric that evaluates the trade-off between return and risk. In a portfolio con-

text, let

$$R_p = \sum_{i=1}^N w_i \mu_i$$

denote the expected log-return of the portfolio, and let

$$\sigma_p = \sqrt{\sum_{i=1}^N \sum_{j=1}^N w_i w_j \sigma_{ij}^2}$$

denote the standard deviation of portfolio log-returns, where σ_{ij} are the entries of the covariance matrix Σ . Let r_f denote the risk-free rate. The Sharpe ratio is then defined as:

$$SR = \frac{R_p - r_f}{\sigma_p}. \quad (\text{A.3})$$

The Maximum Sharpe Ratio portfolio strategy seeks to identify the portfolio weights that maximize this ratio. The resulting optimization problem is given by:

$$\begin{aligned} \max_w \quad & \left(\frac{\sum_{i=1}^N w_i \mu_i - r_f}{\sqrt{\sum_{i=1}^N \sum_{j=1}^N w_i w_j \sigma_{ij}}} \right) = \left(\frac{\mu^T w - r_f}{\sqrt{w^T \Sigma w}} \right), \\ \text{s.t.} \quad & \sum_{i=1}^N w_i = 1, \\ & 0 \leq w_i \leq 1, \forall i = 1, \dots, N. \end{aligned} \quad (\text{A.4})$$

A.3 Asset Allocation Results

The following tables present the results obtained for the portfolio strategies when considering 10, 20, 40, and 80 of the most liquid stocks for asset allocation. The best results are highlighted in bold. Notably, for daily log-returns, cumulative log-returns, and Sharpe ratio, higher values are considered better, while for daily standard deviation, excess kurtosis, and skewness, values closer to 0 are preferred. Additionally, daily log-returns, daily standard deviation, cumulative log-returns, and Sharpe ratio results are multiplied by 100 for easier comparison.

Table 11: Results obtained for the portfolio strategies using the 10 most liquid stocks of the S&P 500 from 10/07/2010 to 10/07/2023. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network GMV	0.0614	1.163	6.401	-0.353	200.69	5.28
	Network MS	0.0819	1.539	1.753	-0.406	267.72	5.32
	Network MV	0.0624	1.171	6.199	-0.355	204.18	5.33
	Sig GMV	0.0560	1.211	5.498	-0.404	183.09	4.62
	Sig MS	0.0886	1.787	0.771	-0.399	289.50	4.95
	Sig MV	0.0561	1.218	5.324	-0.406	183.51	4.61
0.292	Network GMV	0.0617	1.173	5.906	-0.394	201.83	5.26
	Network MS	0.0809	1.532	1.543	-0.374	264.47	5.28
	Network MV	0.0627	1.181	5.666	-0.389	204.88	5.31
	Sig GMV	0.0595	1.232	5.173	-0.370	194.54	4.83
	Sig MS	0.0911	1.794	0.668	-0.380	297.76	5.07
	Sig MV	0.0597	1.239	4.995	-0.373	195.34	4.82
0.321	Network GMV	0.0656	1.173	5.705	-0.436	214.39	5.59
	Network MS	0.0813	1.459	1.893	-0.418	265.72	5.57
	Network MV	0.0666	1.181	5.497	-0.432	217.71	5.64
	Sig GMV	0.0556	1.236	4.815	-0.365	181.82	4.50
	Sig MS	0.0882	1.795	0.582	-0.375	288.40	4.91
	Sig MV	0.0556	1.243	4.680	-0.368	181.89	4.47
0.401	Network GMV	0.0651	1.211	5.876	-0.489	212.90	5.37
	Network MS	0.0790	1.467	2.391	-0.453	258.23	5.38
	Network MV	0.0662	1.220	5.651	-0.485	216.39	5.42
	Sig GMV	0.0527	1.241	5.821	-0.457	172.21	4.24
	Sig MS	0.0896	1.800	0.776	-0.394	292.88	4.97
	Sig MV	0.0531	1.248	5.695	-0.460	173.51	4.25

Table 12: Results obtained for the portfolio strategies using the 20 most liquid stocks of the S&P 500 from 10/07/2010 to 10/07/2023. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network GMV	0.0251	0.997	13.025	-0.733	82.25	2.52
	Network MS	0.0650	1.412	4.723	-0.588	212.41	4.60
	Network MV	0.0263	1.013	12.332	-0.694	86.23	2.60
	Sig GMV	0.0198	1.007	6.817	-0.593	64.96	1.97
	Sig MS	0.0785	1.676	1.561	-0.462	256.50	4.68
	Sig MV	0.0203	1.026	6.501	-0.563	66.48	1.98
0.292	Network GMV	0.0254	1.008	13.070	-0.840	83.00	2.52
	Network MS	0.0642	1.423	4.849	-0.632	209.81	4.51
	Network MV	0.0269	1.026	12.269	-0.803	88.01	2.62
	Sig GMV	0.0240	1.022	6.491	-0.558	78.55	2.35
	Sig MS	0.0815	1.676	1.437	-0.447	266.43	4.86
	Sig MV	0.0241	1.041	6.074	-0.519	78.83	2.31
0.321	Network GMV	0.0292	1.006	11.817	-0.699	95.62	2.90
	Network MS	0.0690	1.410	3.710	-0.479	225.57	4.89
	Network MV	0.0311	1.026	11.070	-0.664	101.67	3.03
	Sig GMV	0.0248	1.031	6.239	-0.574	81.14	2.40
	Sig MS	0.0812	1.673	1.356	-0.438	265.42	4.85
	Sig MV	0.0250	1.051	5.793	-0.532	81.72	2.37
0.401	Network GMV	0.0370	1.030	11.46	-0.767	121.03	3.59
	Network MS	0.0654	1.355	4.276	-0.567	213.74	4.82
	Network MV	0.0392	1.057	10.645	-0.737	128.32	3.71
	Sig GMV	0.0261	1.058	7.074	-0.601	85.41	2.46
	Sig MS	0.0821	1.691	1.641	-0.449	268.44	4.85
	Sig MV	0.0264	1.082	6.546	-0.558	86.50	2.44

Table 13: Results obtained for the portfolio strategies using the 40 most liquid stocks of the S&P 500 from 10/07/2010 to 10/07/2023. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network GMV	0.0196	0.863	11.295	-0.542	64.05	2.27
	Network MS	0.0583	1.278	9.746	-0.758	190.66	4.56
	Network MV	0.0248	0.910	11.646	-0.570	81.20	2.73
	Sig GMV	0.0284	0.895	8.111	-0.438	93.07	3.18
	Sig MS	0.0628	1.546	5.647	-0.680	205.26	4.06
	Sig MV	0.0282	0.935	7.723	-0.422	92.26	3.01
0.292	Network GMV	0.0187	0.868	11.665	-0.648	61.13	2.15
	Network MS	0.0559	1.273	10.453	-0.825	182.75	4.39
	Network MV	0.0235	0.917	12.074	-0.685	76.94	2.56
	Sig GMV	0.0300	0.905	8.168	-0.510	98.21	3.31
	Sig MS	0.0643	1.544	5.138	-0.670	210.17	4.16
	Sig MV	0.0298	0.949	7.790	-0.496	97.57	3.14
0.321	Network GMV	0.0220	0.873	11.305	-0.646	72.03	2.52
	Network MS	0.0599	1.284	9.580	-0.760	195.93	4.67
	Network MV	0.0274	0.926	11.578	-0.677	89.66	2.96
	Sig GMV	0.0303	0.908	8.142	-0.519	99.27	3.34
	Sig MS	0.0652	1.545	5.218	-0.675	213.21	4.22
	Sig MV	0.0304	0.951	7.786	-0.504	99.32	3.19
0.401	Network GMV	0.0291	0.888	10.351	-0.495	95.18	3.27
	Network MS	0.0631	1.247	6.288	-0.533	206.19	5.05
	Network MV	0.0349	0.951	9.936	-0.543	114.09	3.66
	Sig GMV	0.0353	0.931	8.852	-0.555	115.57	3.79
	Sig MS	0.0674	1.566	5.060	-0.641	220.27	4.30
	Sig MV	0.0353	0.977	8.201	-0.537	115.50	3.61

Table 14: Results obtained for the portfolio strategies using the 80 most liquid stocks of the S&P 500 from 10/07/2010 to 10/07/2023. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network GMV	0.0255	0.845	14.065	-0.710	83.54	3.02
	Network MS	0.0440	1.239	9.475	-0.647	143.77	3.54
	Network MV	0.0315	0.948	14.011	-0.849	103.14	3.32
	Sig GMV	0.0323	0.889	12.710	-0.684	105.82	3.64
	Sig MS	0.0583	1.520	4.428	-0.672	190.77	3.84
	Sig MV	0.0292	0.958	11.713	-0.761	95.50	3.05
0.292	Network GMV	0.0247	0.848	13.946	-0.728	80.75	2.91
	Network MS	0.0440	1.232	9.874	-0.643	143.91	3.57
	Network MV	0.0308	0.957	13.844	-0.847	100.92	3.22
	Sig GMV	0.0339	0.896	12.677	-0.676	110.94	3.78
	Sig MS	0.0603	1.528	4.180	-0.651	197.15	3.94
	Sig MV	0.0307	0.967	11.337	-0.718	100.60	3.18
0.321	Network GMV	0.0260	0.854	13.988	-0.791	85.07	3.04
	Network MS	0.0477	1.240	9.846	-0.691	156.02	3.85
	Network MV	0.0331	0.967	13.261	-0.887	108.36	3.42
	Sig GMV	0.0354	0.900	12.420	-0.696	115.75	3.93
	Sig MS	0.0602	1.538	4.151	-0.651	196.79	3.91
	Sig MV	0.0324	0.971	11.148	-0.727	105.93	3.33
0.401	Network GMV	0.0292	0.875	12.574	-0.773	95.68	3.34
	Network MS	0.0473	1.235	7.527	-0.567	154.60	3.83
	Network MV	0.0382	0.998	10.806	-0.809	124.88	3.82
	Sig GMV	0.0374	0.930	12.273	-0.755	122.36	4.02
	Sig MS	0.0585	1.561	3.579	-0.650	191.12	3.74
	Sig MV	0.0335	1.003	10.850	-0.851	109.49	3.34

Table 15: Results obtained for the portfolio strategies with short selling using the 10 most liquid stocks of the S&P 500 from 10/07/2010 to 10/07/2023. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network GMV	0.0620	1.165	9.389	-0.369	202.70	5.32
	Network MS	0.0854	1.675	4.846	-0.293	279.21	5.09
	Network MV	0.0630	1.297	22.013	0.033	206.05	4.86
	Sig GMV	0.0571	1.230	7.847	-0.359	186.79	4.64
	Sig MS	0.1040	2.265	4.506	-0.230	339.87	4.59
	Sig MV	0.0568	1.308	11.103	-0.473	185.73	4.34
0.292	Network GMV	0.0621	1.174	5.905	-0.407	203.10	5.29
	Network MS	0.0851	1.646	1.112	-0.278	278.10	5.16
	Network MV	0.0631	1.305	18.303	-0.0003	204.88	4.84
	Sig GMV	0.0597	1.251	4.522	-0.322	195.29	4.77
	Sig MS	0.1012	2.363	1.946	-0.346	330.88	4.28
	Sig MV	0.0594	1.329	7.559	-0.437	194.163	4.47
0.321	Network GMV	0.0656	1.175	5.691	-0.447	215.72	5.61
	Network MS	0.0813	1.574	1.398	-0.285	281.91	5.48
	Network MV	0.0666	1.306	18.144	-0.033	219.02	5.13
	Sig GMV	0.0543	1.256	4.160	-0.332	177.49	4.32
	Sig MS	0.0991	2.474	4.281	-0.554	323.95	4.00
	Sig MV	0.0539	1.334	7.255	-0.445	176.23	4.04
0.401	Network GMV	0.0658	1.213	5.856	-0.499	215.13	5.42
	Network MS	0.0825	1.566	2.146	-0.339	269.58	5.26
	Network MV	0.0668	1.342	16.748	-0.101	218.51	4.98
	Sig GMV	0.0515	1.263	5.042	-0.410	168.40	4.07
	Sig MS	0.0991	2.474	4.281	-0.554	323.95	4.00
	Sig MV	0.0510	1.340	7.870	-0.510	166.63	3.80

Table 16: Results obtained for the portfolio strategies with short selling using the 20 most liquid stocks of the S&P 500 from 10/07/2010 to 10/07/2023. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network GMV	0.0226	0.982	13.821	-0.711	73.91	2.30
	Network MS	0.0850	2.142	3.580	-0.181	277.94	3.97
	Network MV	0.0108	1.157	34.910	-2.089	35.40	0.93
	Sig GMV	0.0098	1.051	3.761	-0.302	32.31	0.94
	Sig MS	0.1954	3.763	1.434	-0.283	638.56	5.19
	Sig MV	0.0071	1.2236	15.744	-0.210	23.42	0.58
0.292	Network GMV	0.0227	0.991	14.456	-0.868	74.17	2.29
	Network MS	0.0807	2.069	3.382	-0.180	263.85	3.90
	Network MV	0.0112	1.165	34.268	-2.141	36.60	0.96
	Sig GMV	0.0129	1.060	2.760	-0.213	42.23	1.21
	Sig MS	0.2011	3.892	1.583	-0.275	657.21	5.168
	Sig MV	0.0100	1.234	14.742	-0.158	32.90	0.811
0.321	Network GMV	0.0268	0.9877	12.939	-0.708	87.66	2.71
	Network MS	0.0790	2.038	3.123	-0.210	258.15	3.87
	Network MV	0.0152	1.164	33.652	-2.051	49.94	1.31
	Sig GMV	0.0105	1.068	2.586	-0.247	34.57	0.99
	Sig MS	0.1932	3.870	2.505	-0.269	631.23	4.99
	Sig MV	0.0077	1.243	14.190	-0.172	25.30	0.62
0.401	Network GMV	0.0372	1.014	12.499	-0.812	121.70	3.67
	Network MS	0.0762	1.680	2.100	-0.207	249.02	4.53
	Network MV	0.0256	1.194	30.711	-2.000	83.85	2.14
	Sig GMV	0.0127	1.092	3.220	-0.266	41.58	1.16
	Sig MS	0.1614	3.611	1.339	-0.425	527.48	4.47
	Sig MV	0.0096	1.266	13.571	-0.190	31.59	0.76

Table 17: Results obtained for the portfolio strategies with short selling using the 40 most liquid stocks of the S&P 500 from 10/07/2010 to 10/07/2023. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network GMV	0.0129	0.856	10.348	-0.419	42.32	1.51
	Network MS	0.0792	2.175	3.485	-0.103	259.03	3.64
	Network MV	0.0068	1.268	120.999	0.299	22.40	0.54
	Sig GMV	0.0031	0.989	2.508	-0.095	10.13	0.31
	Sig MS	0.2375	4.649	1.495	-0.052	775.96	5.10
	Sig MV	0.0177	1.6245	28.537	-0.246	58.14	1.09
0.292	Network GMV	0.0106	0.862	10.918	-0.556	34.84	1.23
	Network MS	0.0741	2.130	3.020	-0.121	242.25	3.48
	Network MV	0.0045	1.273	119.26	0.247	14.74	0.35
	Sig GMV	0.0036	0.989	1.894	-0.081	11.87	0.36
	Sig MS	0.2346	4.404	0.001	-0.125	766.73	5.32
	Sig MV	0.0188	1.628	28.143	-0.244	61.43	1.15
0.321	Network GMV	0.0144	0.865	11.058	-0.597	47.26	1.67
	Network MS	0.0713	2.062	3.116	-0.163	232.99	3.45
	Network MV	0.0085	1.277	117.76	0.227	28.08	0.67
	Sig GMV	0.0031	0.989	1.907	-0.089	10.23	0.31
	Sig MS	0.2329	4.493	-0.205	-0.112	760.96	5.18
	Sig MV	0.0181	1.63	28.019	-0.245	59.32	1.11
0.401	Network GMV	0.0237	0.867	11.259	-0.494	77.45	2.73
	Network MS	0.0713	1.675	2.504	-0.121	233.21	4.26
	Network MV	0.0183	1.289	113.30	0.230	59.79	1.41
	Sig GMV	0.0065	0.980	2.208	-0.115	21.36	0.66
	Sig MS	0.2183	4.303	-0.535	-0.145	713.21	5.07
	Sig MV	0.0219	1.628	28.185	-0.256	71.79	1.34

Table 18: Results obtained for the portfolio strategies with short selling using the 80 most liquid stocks of the S&P 500 from 10/07/2010 to 10/07/2023. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network GMV	0.0174	0.863	9.502	-0.358	56.88	2.01
	Network MS	0.0599	2.354	1.547	-0.114	196.01	2.54
	Network MV	0.0242	1.708	109.758	1.390	79.37	1.42
	Sig GMV	0.0129	1.019	4.648	-0.191	42.21	1.26
	Sig MS	0.3075	7.112	7.252	0.057	1004.80	4.32
	Sig MV	0.0724	3.209	20.512	0.484	236.57	2.25
0.292	Network GMV	0.0159	0.864	10.280	-0.382	52.23	1.84
	Network MS	0.0566	2.285	1.547	-0.117	185.10	2.47
	Network MV	0.0223	1.709	109.398	1.381	72.90	1.30
	Sig GMV	0.0140	1.026	4.665	-0.208	45.95	1.37
	Sig MS	0.3157	7.293	9.430	0.281	1031.55	4.32
	Sig MV	0.0748	3.221	20.154	0.475	244.51	2.32
0.321	Network GMV	0.0175	0.861	10.489	-0.432	57.31	2.03
	Network MS	0.0536	2.193	2.106	-0.133	175.25	2.44
	Network MV	0.0237	1.710	109.185	1.366	77.73	1.39
	Sig GMV	0.0132	1.027	4.751	-0.203	43.15	1.28
	Sig MS	0.2870	7.158	7.551	0.129	937.79	4.00
	Sig MV	0.0739	3.228	19.942	0.475	241.65	2.29
0.401	Network GMV	0.0225	0.860	12.206	-0.508	73.72	2.62
	Network MS	0.0506	1.974	4.134	-0.119	165.58	2.56
	Network MV	0.0309	1.727	104.915	1.303	101.16	1.79
	Sig GMV	0.0166	1.020	4.028	-0.259	54.44	1.63
	Sig MS	0.2854	7.175	2.196	-0.162	932.61	3.97
	Sig MV	0.0776	3.221	20.141	0.469	253.81	2.41

Table 19: Results for the out-of-sample portfolio strategies with and without short selling using the 10 most liquid stocks of the S&P 500 from 11/07/2023 to 31/01/2024. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network MS	0.0486	1.266	7.562	-1.77	6.70	3.83
	Sig MS	0.0026	1.643	10.701	-2.542	0.36	0.15
	Network MS (short)	-0.1015	2.062	3.339	-1.576	-14.01	-4.92
	Sig MS (short)	0.1229	2.685	5.403	1.128	16.96	4.57
0.292	Network MS	0.0492	1.265	7.532	-1.769	6.80	3.89
	Sig MS	0.0026	1.643	10.701	-2.542	0.36	0.15
	Network MS (short)	-0.0958	2.014	3.185	-1.587	-13.22	-4.75
	Sig MS (short)	0.1229	2.685	5.403	1.128	16.96	4.57
0.321	Network MS	0.0520	1.269	7.410	-1.756	7.17	4.09
	Sig MS	0.0026	1.643	10.701	-2.542	0.36	0.15
	Network MS (short)	-0.1010	2.014	3.181	-1.591	-13.93	-5.01
	Sig MS (short)	0.1229	2.685	5.403	1.128	16.96	4.57
0.401	Network MS	0.0522	1.273	6.844	-1.671	7.20	4.10
	Sig MS	0.0026	1.643	10.701	-2.542	0.36	0.15
	Network MS (short)	-0.1058	1.999	3.275	-1.595	-14.61	-5.29
	Sig MS (short)	0.1229	2.685	5.403	1.128	16.96	4.57

Table 20: Results for the out-of-sample portfolio strategies with and without short selling using the 20 most liquid stocks of the S&P 500 from 11/07/2023 to 31/01/2024. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network MS	0.0216	1.083	9.799	-2.269	2.98	1.99
	Sig MS	0.0133	1.770	7.530	-2.028	1.84	0.75
	Network MS (short)	-0.1645	2.789	3.661	-1.012	-22.70	-5.89
	Sig MS (short)	0.0521	2.738	-0.262	-0.389	7.20	1.90
0.292	Network MS	0.0166	1.068	10.439	-2.304	2.29	1.55
	Sig MS	0.0133	1.770	7.530	-2.028	1.84	0.75
	Network MS (short)	-0.2065	2.770	0.873	-1.046	-28.49	-7.45
	Sig MS (short)	0.0521	2.738	-0.262	-0.389	7.20	1.90
0.321	Network MS	0.0148	1.063	11.279	-2.389	2.05	1.39
	Sig MS	0.0133	1.770	7.530	-2.028	1.84	0.75
	Network MS (short)	-0.2273	2.792	1.387	-1.061	-31.37	-8.14
	Sig MS (short)	0.0521	2.738	-0.262	-0.389	7.20	1.90
0.401	Network MS	0.0091	1.036	9.533	-2.219	1.25	0.87
	Sig MS	0.0133	1.770	7.530	-2.028	1.84	0.75
	Network MS (short)	-0.2261	2.925	1.655	-0.911	-31.20	-7.72
	Sig MS (short)	0.0521	2.738	-0.262	-0.389	7.20	1.90

Table 21: Results for the out-of-sample portfolio strategies with and without short selling using the 40 most liquid stocks of the S&P 500 from 11/07/2023 to 31/01/2024. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network MS	0.0433	0.7495	-1.624	-0.117	5.97	5.77
	Sig MS	0.0985	1.500	19.893	-0.583	13.60	6.56
	Network MS (short)	-0.5542	3.197	3.235	-1.598	-76.49	-17.33
	Sig MS (short)	-0.3375	3.440	1.229	-0.773	-46.57	-9.80
0.292	Network MS	0.0401	0.724	-1.660	-0.128	5.54	5.54
	Sig MS	0.0985	1.500	19.893	-0.583	13.60	6.56
	Network MS (short)	-0.5396	3.040	3.432	-1.584	-74.46	-17.74
	Sig MS (short)	-0.3375	3.440	1.229	-0.773	-46.57	-9.80
0.321	Network MS	0.0438	0.722	-1.570	-0.099	6.05	6.07
	Sig MS	0.0985	1.500	19.893	-0.583	13.60	6.56
	Network MS (short)	-0.5968	3.231	3.682	-1.662	-82.36	-18.47
	Sig MS (short)	-0.3375	3.440	1.229	-0.773	-46.57	-9.80
0.401	Network MS	0.0319	0.721	-1.715	-0.071	4.40	4.42
	Sig MS	0.0985	1.500	19.893	-0.583	13.60	6.56
	Network MS (short)	-0.5687	3.220	3.386	-1.481	-78.48	-17.66
	Sig MS (short)	-0.3375	3.440	1.229	-0.773	-46.57	-9.80

Table 22: Results for the out-of-sample portfolio strategies with and without short selling using the 80 most liquid stocks of the S&P 500 from 11/07/2023 to 31/01/2024. Best results are shown in bold.

c_θ	Strategy	Daily Mean	Daily Std	Excess Kurt.	Skew.	Cumul. Log-Rets	SR
0.248	Network MS	0.0625	0.763	-2.534	-0.002	8.63	8.19
	Sig MS	0.1224	0.941	-1.519	-0.486	16.89	13.00
	Network MS (short)	-0.3358	6.677	-2.118	-0.610	-46.34	-5.02
	Sig MS (short)	0.1158	0.949	-1.161	-0.393	15.98	12.20
0.292	Network MS	0.0604	0.756	-2.471	-0.022	8.34	7.98
	Sig MS	0.1224	0.941	-1.519	-0.486	16.89	13.00
	Network MS (short)	-0.3568	6.694	-2.075	-0.566	-49.24	-5.33
	Sig MS (short)	0.1158	0.949	-1.161	-0.393	15.98	12.20
0.321	Network MS	0.0617	0.745	-2.495	0.015	8.52	8.28
	Sig MS	0.1224	0.941	-1.519	-0.486	16.89	13.00
	Network MS (short)	0.4070	6.615	-2.112	-0.600	-56.16	-6.15
	Sig MS (short)	0.1158	0.949	-1.161	-0.393	15.98	12.20
0.401	Network MS	0.0607	0.745	-2.357	0.020	8.38	8.15
	Sig MS	0.1224	0.941	-1.519	-0.486	16.89	13.00
	Network MS (short)	0.5001	6.549	-1.796	-0.765	-69.02	-7.63
	Sig MS (short)	0.1158	0.949	-1.161	-0.393	15.98	12.20

Appendix B

Time Series Forecasting using Geometric Deep Learning

B.1 Evaluation Metrics

The following standard evaluation metrics: Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) are defined as follows:

$$\text{MSE} = \frac{1}{l} \sum_{i=1}^l (y_i - \hat{y}_i)^2, \quad (\text{B.1})$$

$$\text{RMSE} = \sqrt{\frac{1}{l} \sum_{i=1}^l (y_i - \hat{y}_i)^2}, \quad (\text{B.2})$$

$$\text{MAE} = \frac{1}{l} \sum_{i=1}^l |y_i - \hat{y}_i|, \quad (\text{B.3})$$

$$\text{MAPE} = \frac{1}{l} \sum_{i=1}^l \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \quad (\text{B.4})$$

where l denotes the number of samples in the corresponding dataset split, y_i is the true target value, and $\hat{y}_i$ is the model prediction.

B.2 Hyperparameters

This section describes the hyperparameter optimization procedures used in the proposed analysis. We first detail the optimization strategy for the baseline models and subsequently discuss the hyperparameters associated with the Time-Geometric model.

Hyperparameter optimization is performed using the Optuna Python framework (Akiba et al., 2019), which provides efficient and flexible support for tuning a wide range of machine learning models. Optuna is based on a *study-trial* paradigm, where a *study* corresponds to an optimization process defined by an objective function, and a *trial* represents a single evaluation of a candidate set of hyperparameters. The objective of each study is to identify an optimal configuration through repeated trials. In our experiments, we conduct 1,000 trials for each model under consideration. Finally, we note that the loss function, regularization term, and optimization algorithm are fixed *a priori* in order to ensure computational tractability and comparability across models.

B.2.1 Baseline Models

The baseline models considered in this study include Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), Recurrent Neural Network (RNN), Bidirectional RNN (BiRNN), Gated Recurrent Unit (GRU), Bidirectional GRU (BiGRU), Transformer, and Temporal Convolutional Network (TCN). The hyperparameters optimized for these architectures comprise the learning rate, number of neurons, number of layers, sequence length, batch size, dropout rate, and activation function. Detailed configurations for the baseline models corresponding to prediction horizons of 1, 5, and 20 days are reported in Tables 23, 24, and 25, respectively.

Table 23: The table displays the hyperparameter values for a 1 day prediction and the baseline models considered.

HYPERPARAM.	LSTM	BiLSTM	RNN	BiRNN	GRU	BiGRU	TRANF.	TCN
LEARNING RATE	0.000176	0.000258	0.000985	0.000656	0.000405	0.000134	0.000883	0.000935
# NEURONS	190	140	110	30	120	30	-	200
# LAYERS	2	2	2	2	3	2	2	-
SEQUENCE LENGTH	100	100	100	100	100	100	100	100
BATCH SIZE	40	160	160	80	160	80	40	160
DROPOUT RATE	0.08	0.47	0.06	0.03	0.33	0.06	0.01	0.08
ACTIVATION FUNCTION	SELU	SELU	SELU	SELU	ELU	SELU	ELU	LEAKY RELU

Table 24: The table displays the hyperparameter values for a 5 day prediction and the baseline models considered.

HYPERPARAM.	LSTM	BiLSTM	RNN	BiRNN	GRU	BiGRU	TRANF.	TCN
LEARNING RATE	0.000731	0.000731	0.000168	5.254E-05	0.000165	0.000572	0.000884	8.053E-05
# NEURONS	190	120	140	190	50	180	-	50
# LAYERS	4	6	2	5	4	2	2	-
SEQUENCE LENGTH	100	100	100	100	100	100	100	100
BATCH SIZE	160	160	160	160	160	160	80	120
DROPOUT RATE	0.29	0.37	0.32	0.23	0.36	0.23	0.27	0.41
ACTIVATION FUNCTION	SELU	ELU	SELU	SELU	PRELU	PRELU	LEAKY RELU	SELU

B.2.2 Time-Geometric Model

For the Time Component of the Time-Geometric model, we retain the same hyperparameter configurations as those identified for the corresponding baseline models, as reported in Tables 23, 24, and 25 for prediction horizons of 1, 5, and 20 days. The Geometric Component introduces additional hyperparameters, including the dropout rate, the number of layers and neurons in the Graph Convolutional Network, the number of neurons in the LSTM layer, the inclusion of dropout, and the choice of graph directionality. Specifically, we consider either an undirected graph (denoted as “None”), or a directed graph with left-to-right temporal orientation (denoted as “L-TO-R”). In contrast, the Fully Connected Component involves a more limited set of hyperparameters, namely the inclusion of a skip connection and the activation function, which is set—without loss of generality—to match the activation function used in the corresponding baseline model. The complete set of optimized hyperparameters for the Time-Geometric model, for prediction

Table 25: The table displays the hyperparameter values for a 20 day prediction and the baseline models considered.

HYPERPARAM.	LSTM	BiLSTM	RNN	BiRNN	GRU	BiGRU	TRANF.	TCN
LEARNING RATE	0.000899	5.518E-05	0.000818	8.943E-05	0.000371	0.000117	0.000206	0.000178
# NEURONS	180	60	70	20	180	140	-	30
# LAYERS	6	5	3	10	6	8	2	-
SEQUENCE LENGTH	100	100	100	100	100	100	100	100
BATCH SIZE	80	160	80	160	160	80	160	160
DROPOUT RATE	0.25	0.49	0.28	0.22	0.42	0.19	0.30	0.36
ACTIVATION FUNCTION	SELU	ELU	ELU	RELU	SELU	ELU	LEAKY RELU	ELU

horizons of 1, 5, and 20 days, is summarized in Tables 26, 27, and 28, respectively.

Table 26: The table displays the hyperparameter values for a 1 day prediction and the Time-Geometric model, denoted as TG, with the Time Component model values enclosed in brackets.

HYPERPARAM.	TG(LSTM)	TG(BiLSTM)	TG(RNN)	TG(BiRNN)	TG(GRU)	TG(BiGRU)	TG(TRANF.)	TG(TCN)
DROPOUT RATE	0.37	0.02	0.22	0.48	0.33	0.05	0.37	0.24
# NEURONS	40	190	110	90	160	120	110	60
# LAYER	9	9	7	7	8	10	2	7
# NEURONS LSTM	40	20	100	30	180	30	180	80
DROPOUT?	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE
SKIP LAYER	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE
DIRECTION	L-TO-R	NONE	L-TO-R	NONE	L-TO-R	L-TO-R	NONE	NONE

Table 27: The table displays the hyperparameter values for a 5 day prediction and the Time-Geometric model, denoted as TG, with the Time Component model values enclosed in brackets.

HYPERPARAM.	TG(LSTM)	TG(BiLSTM)	TG(RNN)	TG(BiRNN)	TG(GRU)	TG(BiGRU)	TG(TranF.)	TG(TCN)
DROPOUT RATE	0.25	0.17	0.13	0.4	0.06	0.03	0.03	0.01
# NEURONS	140	80	160	180	30	100	150	160
# LAYER	5	7	10	2	6	7	5	10
# NEURONS LSTM	160	170	20	130	190	100	130	80
DROPOUT?	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
SKIP LAYER	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE
DIRECTION	L-TO-R	NONE	L-TO-R	NONE	NONE	L-TO-R	NONE	NONE

Table 28: The table displays the hyperparameter values for a 20 day prediction and the Time-Geometric model, denoted as TG, with the Time Component model values enclosed in brackets.

HYPERPARAM.	TG(LSTM)	TG(BiLSTM)	TG(RNN)	TG(BiRNN)	TG(GRU)	TG(BiGRU)	TG(TranF.)	TG(TCN)
DROPOUT RATE	0.21	0.15	0.11	0.13	0.05	0.45	0.08	0.29
# NEURONS	130	160	110	160	190	20	90	180
# LAYER	5	2	6	5	9	2	2	3
# NEURONS LSTM	50	20	190	170	80	20	190	50
DROPOUT?	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
SKIP LAYER	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
DIRECTION	NONE	NONE	NONE	NONE	L-TO-R	NONE	NONE	L-TO-R

B.3 Empirical Results

This section presents the complete set of results and visualizations obtained from the statistical significance analysis conducted to compare the Time-Geometric model with the various baseline models considered in this study. It is important to note that a smaller value indicates better performance relative to the others. The best-performing model for each metric is highlighted in bold.

Table 29: The table displays the average metric values for a 1 day prediction across all models in the considered dataset. In parentheses, we specify the Time Component considered in the Time-Geometric model (TG). The best-performing model for each metric is highlighted in bold.

Model	RMSE	MAE	MAPE	MASE
LSTM	2.0743	1.2570	0.9074	0.5810
BiLSTM	2.1349	1.2596	0.9276	0.5957
RNN	2.2012	1.3841	1.0110	0.6520
BiRNN	2.0006	1.1962	0.8797	0.5734
GRU	4.6464	2.7335	2.1174	1.3166
BiGRU	2.0438	1.1416	0.8441	0.5422
Transf.	6.5977	4.4957	3.2390	2.0907
TCN	5.7037	4.2612	2.8537	1.8767
TG(LSTM)	2.0580	1.1044	0.8262	0.5248
TG(BiLSTM)	2.1141	1.1311	0.8487	0.5338
TG(RNN)	2.2554	1.4192	1.0122	0.6553
TG(BiRNN)	2.0386	1.1529	0.8419	0.5377
TG(GRU)	4.6131	2.5326	1.9950	1.2150
TG(BiGRU)	2.1665	1.1420	0.8570	0.5360
TG(Transf.)	4.5617	2.5229	1.9869	1.2190
TG(TCN)	1.1420	0.8376	0.5658	0.3828

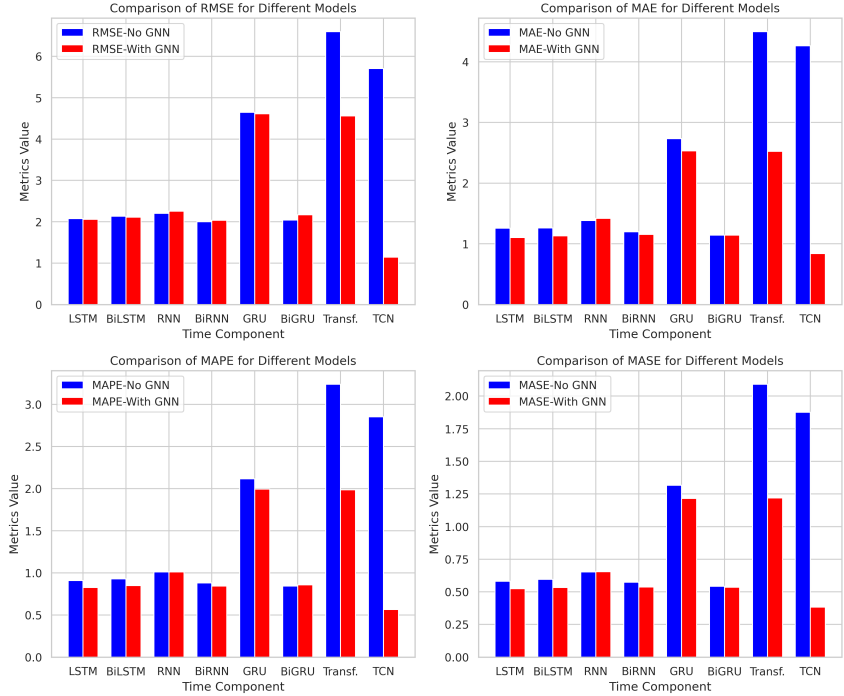


Figure 19: Average metric values for 1 day prediction for all the models within the considered dataset are presented in the figure. The x-axis denotes the models used both as baselines and in the Time Component. The blue bar represents the average metric values for the respective baseline model, while the red bar signifies the average metric value for the Time-Geometric model.

Table 30: The table displays the average metric values for a 5 day prediction across all models in the considered dataset. In parentheses, we specify the Time Component considered in the Time-Geometric model (TG). The best-performing model for each metric is highlighted in bold.

Model	RMSE	MAE	MAPE	MASE
LSTM	4.0979	3.0234	1.9594	1.3064
BiLSTM	5.7802	4.2239	3.1294	2.0567
RNN	3.1201	2.3216	1.5447	1.0180
BiRNN	3.0803	2.2928	1.5426	1.0238
GRU	2.6417	2.0466	1.2799	0.8665
BiGRU	3.0593	2.3167	1.4362	0.9515
Transf.	9.4793	7.1196	4.8149	3.1075
TCN	8.3543	6.2987	4.3513	2.8774
TG(LSTM)	2.5442	1.7810	1.2149	0.7979
TG(BiLSTM)	4.4332	2.8562	2.1135	1.3274
TG(RNN)	2.6913	1.8169	1.2785	0.8272
TG(BiRNN)	3.1389	2.3015	1.5676	1.0375
TG(GRU)	2.1209	1.6391	1.0352	0.6921
TG(BiGRU)	2.7205	2.0417	1.2581	0.8224
TG(Transf.)	2.3148	1.7685	1.1219	0.7554
TG(TCN)	3.5292	2.5441	1.8232	1.1920

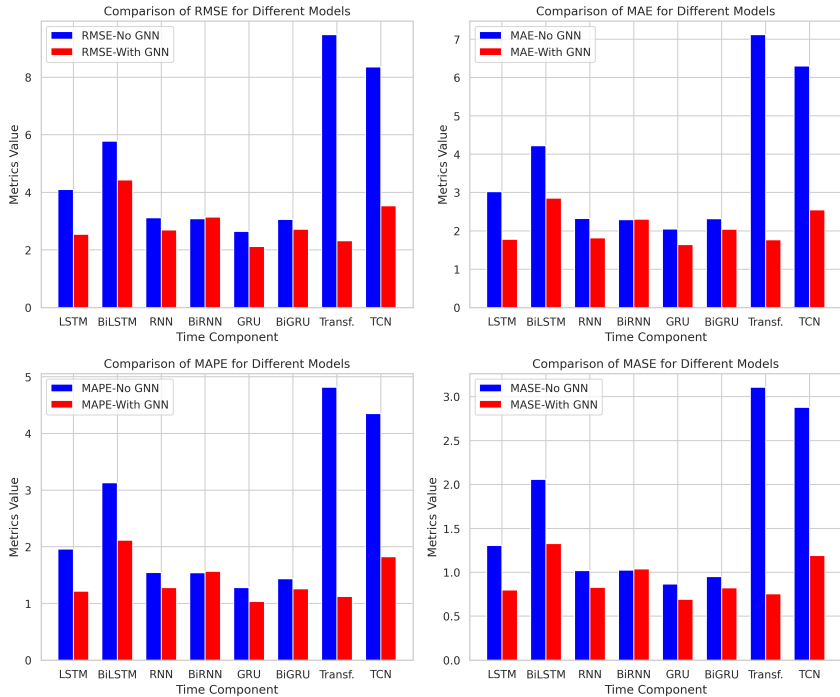


Figure 20: Average metric values for 5 day prediction for all the models within the considered dataset are presented in the figure. The x-axis denotes the models used both as baselines and in the Time Component. The blue bar represents the average metric values for the respective baseline model, while the red bar signifies the average metric value for the Time-Geometric model.

Table 31: The table displays the average metric values for a 20 day prediction across all models in the considered dataset. In parentheses, we specify the Time Component considered in the Time-Geometric model (TG). The best-performing model for each metric is highlighted in bold.

Model	RMSE	MAE	MAPE	MASE
LSTM	11.3770	9.0394	6.1697	4.0384
BiLSTM	7.2682	5.7614	3.9615	2.5693
RNN	7.7205	6.1167	3.9599	2.6271
BiRNN	9.5645	7.4786	5.1974	3.3012
GRU	7.6971	6.1448	4.0830	2.7182
BiGRU	8.2659	6.5398	4.2292	2.8106
Transf.	11.0544	8.7913	6.1235	3.9174
TCN	11.8729	8.9351	5.9946	3.8302
TG(LSTM)	12.5783	10.0061	6.7539	4.2900
TG(BiLSTM)	5.6507	4.3606	2.9326	1.9024
TG(RNN)	8.1585	6.5460	4.3116	2.8110
TG(BiRNN)	9.1312	6.9099	4.9102	3.0900
TG(GRU)	8.8487	7.1136	4.6734	3.0290
TG(BiGRU)	8.4649	6.6535	4.3596	2.9185
TG(Transf.)	10.7575	8.5082	5.8595	3.7253
TG(TCN)	12.6266	9.5592	6.2406	3.9679

Table 32: The table presents the outcomes of the tests, with Acceptance denoted as *A* and Rejection as *R*. The test statistics for the Paired t-test, Wilcoxon test, and Sign test are encapsulated in brackets. The entry *Δ-“Model”* signifies the comparison between the baseline model and its corresponding Time-Geometric model. For the Paired t-test, we consider 89 degrees of freedom, resulting in a critical value of 1.986. In the Wilcoxon test, the critical value is 1.9599, and for the Sign test, the critical value is 54.42.

MODEL	PAIRED T-TEST				WILCOXON TEST				SIGN TEST			
	RMSE	MAE	MAPE	MASE	RMSE	MAE	MAPE	MASE	RMSE	MAE	MAPE	MASE
Δ LSTM	R(6.0)	R(6.6)	R(9.4)	R(11)	R(8.0)	R(8.1)	R(8.1)	R(8.0)	R(87)	R(88)	R(88)	R(88)
Δ BiLSTM	R(4.4)	R(5.6)	R(6.5)	R(6.4)	R(7.6)	R(7.9)	R(8.1)	R(8.1)	R(84)	R(86)	R(86)	R(86)
Δ RNN	R(4.4)	R(5.6)	R(9.6)	R(8.1)	R(5.9)	R(7.3)	R(7.4)	R(7.1)	R(73)	R(80)	R(80)	R(80)
Δ BiRNN	A(1.2)	A(0.2)	A(0.8)	A(1.1)	R(2.6)	A(0.6)	A(1.0)	A(1.2)	A(30)	A(44)	A(44)	A(43)
Δ GRU	R(4.4)	R(4.4)	R(8.6)	R(5.9)	R(6.7)	R(6.4)	R(6.6)	R(6.4)	R(76)	R(76)	R(76)	R(75)
Δ BiGRU	R(2.1)	R(2.2)	R(2.1)	A(1.9)	R(3.8)	R(3.6)	R(3.6)	R(3.5)	R(60)	R(58)	R(58)	R(57)
Δ TRANSF.	R(7.1)	R(8.0)	R(18)	R(12)	R(8.2)	R(8.2)	R(8.2)	R(8.2)	R(90)	R(90)	R(90)	R(90)
Δ TCN	R(6.1)	R(7.2)	R(16)	R(11)	R(8.2)	R(8.2)	R(8.2)	R(8.2)	R(90)	R(90)	R(90)	R(90)

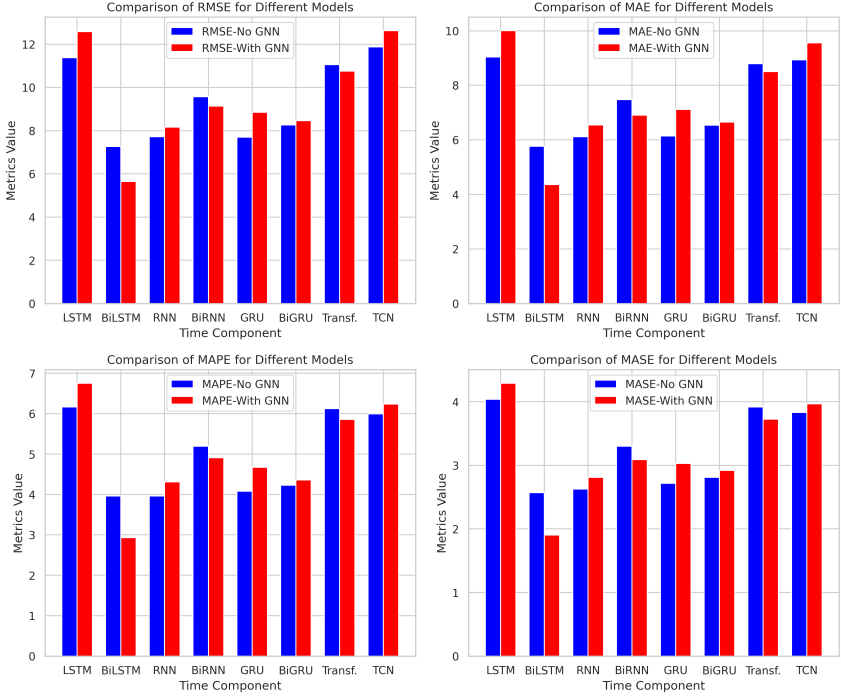


Figure 21: Average metric values for 20 day prediction for all the models within the considered dataset are presented in the figure. The x-axis denotes the models used both as baselines and in the Time Component. The blue bar represents the average metric values for the respective baseline model, while the red bar signifies the average metric value for the Time-Geometric model.

Table 33: The table presents the outcomes of the tests, with Acceptance denoted as A and Rejection as R . The test statistics for the Paired t-test, Wilcoxon test, and Sign test are encapsulated in brackets. The entry Δ -“Model” signifies the comparison between the baseline model and its corresponding Time-Geometric model. For the Paired t-test, we consider 89 degrees of freedom, resulting in a critical value of 1.986. In the Wilcoxon test, the critical value is 1.9599, and for the Sign test, the critical value is 54.42.

MODEL	PAIRED T-TEST				WILCOXON TEST				SIGN TEST			
	RMSE	MAE	MAPE	MASE	RMSE	MAE	MAPE	MASE	RMSE	MAE	MAPE	MASE
Δ LSTM	A(1.4)	A(1.5)	A(1.5)	A(1.7)	A(1.0)	A(1.0)	A(0.9)	A(1.0)	A(42)	A(41)	A(41)	A(40)
Δ BiLSTM	R(8.7)	R(8.8)	R(9.1)	R(7.5)	R(8.2)	R(8.2)	R(8.2)	R(8.2)	R(89)	R(90)	R(90)	R(89)
Δ RNN	A(1.1)	A(1.5)	A(1.98)	R(2.1)	A(1.7)	R(2.1)	R(2.5)	R(2.5)	A(39)	A(39)	A(39)	A(39)
Δ BiRNN	R(4.9)	R(5.7)	R(8.0)	R(8.3)	R(7.2)	R(7.3)	R(7.1)	R(7.2)	R(81)	R(82)	R(82)	R(80)
Δ GRU	R(2.4)	R(2.5)	R(3.3)	R(3.1)	R(2.5)	R(2.5)	R(2.7)	R(2.6)	A(37)	A(35)	A(35)	A(34)
Δ BiGRU	A(0.5)	A(0.4)	A(1.0)	A(0.9)	A(1.2)	A(0.8)	A(0.8)	A(0.7)	A(40)	A(43)	A(43)	A(39)
Δ TRANSF.	A(1.1)	A(1.2)	R(3.1)	R(2.5)	R(4.0)	R(3.9)	R(3.8)	R(3.8)	R(66)	R(66)	R(66)	R(66)
Δ TCN	A(1.1)	A(1.2)	A(1.2)	A(1.0)	A(1.5)	R(2.1)	A(1.8)	A(1.8)	A(40)	A(34)	A(34)	A(33)

Table 34: The table presents the average ranks for the algorithms computed on each dataset. Values in bold indicate the best rank for each metric.

MODEL	1-DAY				5-DAY				20-DAY			
	RMSE	MAE	MAPE	MASE	RMSE	MAE	MAPE	MASE	RMSE	MAE	MAPE	MASE
LSTM	5.71	6.86	6.78	6.86	10.92	11.23	11.05	11.23	12.38	12.40	12.41	12.40
BiLSTM	7.50	8.12	8.17	8.12	13.95	13.94	13.96	13.94	5.10	5.10	5.20	5.10
RNN	8.21	8.78	8.67	8.78	8.52	8.50	8.48	8.50	5.54	5.66	5.55	5.66
BiRNN	4.86	6.46	6.24	6.46	8.63	8.93	8.93	8.93	10.02	10.05	10.15	10.05
GRU	13.74	13.90	13.90	13.90	5.51	6.22	6.05	6.22	5.87	5.93	5.86	5.93
BiGRU	6.05	5.90	5.92	5.90	5.13	5.55	5.52	5.55	6.70	6.82	6.67	6.82
TRANSF.	15.75	15.65	15.67	15.65	15.27	15.34	15.30	15.34	12.47	12.61	12.68	12.61
TCN	14.07	14.92	14.81	14.92	15.18	15.25	15.27	15.25	11.27	10.65	10.54	10.65
TG(LSTM)	5.34	4.57	4.56	4.57	4.90	4.43	4.44	4.43	12.90	13.11	13.08	13.11
TG(BiLSTM)	6.91	5.07	5.24	5.07	12.11	11.21	11.42	11.21	1.50	1.35	1.38	1.35
TG(RNN)	8.07	8.34	8.37	8.34	6.23	5.05	5.20	5.05	6.42	6.64	6.62	6.64
TG(BiRNN)	4.74	5.28	5.32	5.28	9.15	8.77	8.86	8.77	8.36	7.94	8.22	7.94
TG(GRU)	13.60	12.46	12.57	12.46	2.18	2.51	2.44	2.51	7.27	7.43	7.40	7.43
TG(BiGRU)	7.60	5.15	5.28	5.15	3.96	4.26	4.30	4.26	7.21	7.33	7.34	7.33
TG(TRANSF.)	12.56	12.48	12.52	12.48	3.47	3.85	3.78	3.85	11.63	11.70	11.74	11.70
TG(TCN)	1.23	1.97	1.90	1.97	10.82	10.90	10.93	10.90	11.30	11.23	11.08	11.23

Nemenyi Test for Evaluation Metrics with 1 day forecasting horizon

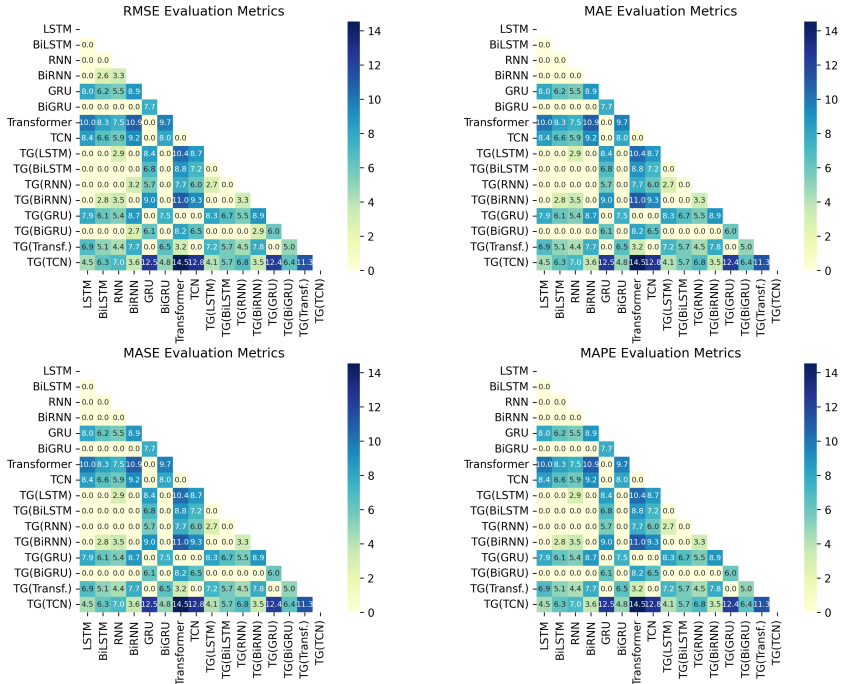


Figure 22: Nemenyi test for 1 day forecasting. Non-statistically significant relationships are highlighted in yellow.

Nemenyi Test for Evaluation Metrics with 5 day forecasting horizon

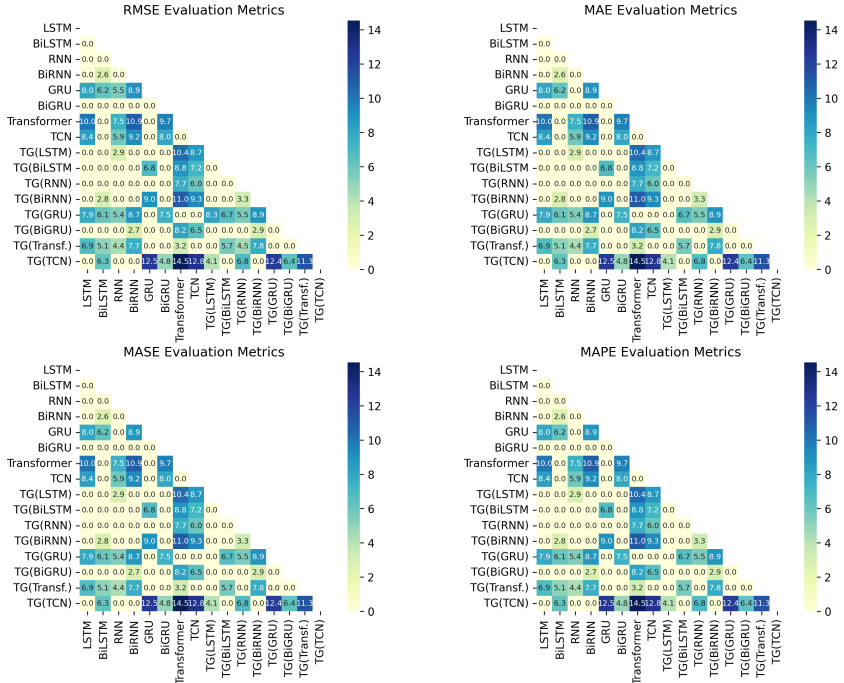


Figure 23: Nemenyi test for 5 day forecasting. Non-statistically significant relationships are highlighted in yellow.

Nemenyi Test for Evaluation Metrics with 20 day forecasting horizon

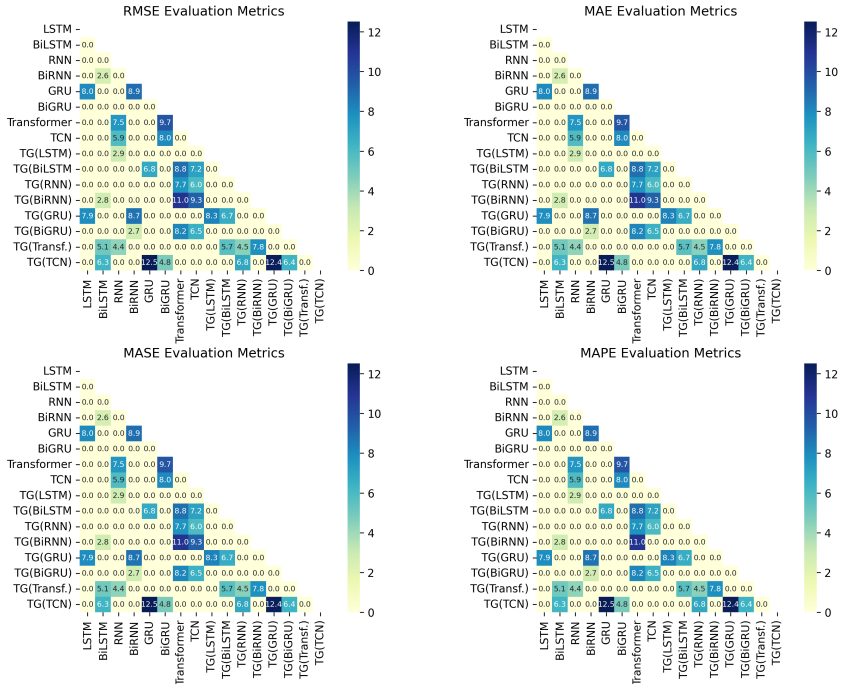


Figure 24: Nemenyi test for 20 day forecasting. Non-statistically significant relationships are highlighted in yellow.

Appendix C

Advances in Geometric Deep Learning

C.1 Baseline Models Hyperparameters

For the empirical analysis, we consider the following baseline models: a Graph Convolutional Network (GCN) augmented with a Long Short-Term Memory (LSTM) layer, an attention-based LSTM model (AT-LSTM), an attention-based Gated Recurrent Unit (AT-GRU), an attention-based bidirectional LSTM (AT-BiLSTM), a convolutional neural network augmented with an attention-based LSTM (CNN-AT-LSTM), a Transformer, and the *Time-Geometric* (TG) model.

The common hyperparameters across all deep learning models include the batch size (BS), the time window size (w), the learning rate (lr), and the dropout rate (Dropout). For the GCN-LSTM model, the specific hyperparameters are: the number of GCN layers (# Layer), the number of neurons in the GCN component (# tg), and the number of neurons in the LSTM layer (# tl). The corresponding optimized hyperparameter values are reported in Table 35.

For the attention-based LSTM model, the specific hyperparameters include: the number of LSTM layers (# Layer), the number of neurons in the LSTM (# tl), and the number of attention heads. The optimized

Table 35: Definition of the hyperparameters for the GCN-LSTM Model.

Dataset	BS	w	lr	Dropout	# Layer	# tg	# tl
Stock Prices	7.545×10^{-6}	0.3	1	50	15	200	6
Energy Consumption	5.815×10^{-6}	0.3	5	15	25	130	2
Air Quality	9.083×10^{-5}	0.3	5	35	35	100	4

Table 36: Definition of the hyperparameters for the AT-LSTM, AT-GRU, AT-BiLSTM, and CNN-AT-LSTM Models.

Dataset	BS	w	lr	Dropout	# Layer	# tl
Stock Prices	7.545×10^{-6}	0.3	1	50	15	200
Energy Consumption	5.815×10^{-6}	0.3	5	15	25	130
Air Quality	9.083×10^{-5}	0.3	5	35	35	100

configurations for this model are provided in Table 36. Note that also AT-GRU, AT-BiLSTM, and CNN-AT-LSTM models have the same specific hyperparameters set and for simplicity we consider the same values.

For the *Time-Geometric* model, the hyperparameters depend on the configuration of both the temporal and geometric components. In this study, we adopt an LSTM architecture for the temporal component and a GCN for the geometric component. Accordingly, the specific hyperparameters include: the number of layers in the LSTM (# Layer LSTM), the number of neurons in the LSTM (# tl), the number of layers in the GCN (# Layer GCN), the number of neurons in the GCN (# tg), and the number of neurons in the fully connected layers (# tc). The optimized hyperparameter values for the TG model are reported in Table 37.

Table 37: Definition of the hyperparameters for the *Time-Geometric* Model.

Dataset	BS	w	lr	Dropout	# Layer LSTM	# tl	# Layer GCN	# tg	# tc
Stock Prices	7.545×10^{-6}	0.3	1	50	15	200	6	32	0.1
Energy Consumption	5.815×10^{-6}	0.3	5	15	25	130	2	32	0.4
Air Quality	9.083×10^{-5}	0.3	5	35	35	100	4	32	0.2

Bibliography

- Abidin, Siti Nazifah Zainol and Maheran Mohd Jaffar (2014). "Forecasting Share Prices of Small Size Companies in Bursa Malaysia using Geometric Brownian Motion". In: *Applied Mathematics & Information Sciences* 8.1, p. 107.
- Adrian, Tobias and Hyun Song Shin (2010). "The Changing Nature of Financial Intermediation and the Financial Crisis of 2007–2009". In: *Annu. Rev. Econ.* 2.1, pp. 603–618.
- Akaike, Hirotugu (1998). "Information Theory and an Extension of the Maximum Likelihood Principle". In: *Selected papers of hirotugu akaike*. Springer, pp. 199–213.
- Akiba, Takuya et al. (2019). "Optuna: A Next-Generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631.
- Amihud, Yakov (2002). "Illiquidity and Stock Returns: Cross-Section and Time-Series Effects". In: *Journal of financial markets* 5.1, pp. 31–56.
- Ansari, Abdul Fatir et al. (2024). "Chronos: Learning the Language of Time Series". In: *Transactions on Machine Learning Research*. <https://openreview.net/forum?id=gerNCVqqtr>. ISSN: 2835-8856.
- Antelmi, Alessia et al. (2023). "A Survey on Hypergraph Representation Learning". In: *ACM Computing Surveys* 56.1, pp. 1–38.
- Ariyo, Adebisi A, Adewumi O Adewumi, and Charles K Ayo (2014). "Stock Price Prediction using the ARIMA Model". In: *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*. IEEE, pp. 106–112.
- Assefa, Samuel A et al. (2020). "Generating Synthetic Data in Finance: Opportunities, Challenges and Pitfalls". In: *Proceedings of the First ACM International Conference on AI in Finance*, pp. 1–8.

- Bai, Song, Feihu Zhang, and Philip HS Torr (2021). "Hypergraph Convolution and Hypergraph Attention". In: *Pattern Recognition* 110, p. 107637.
- Bai, Zhidong and Jack W Silverstein (2010). *Spectral Analysis of Large Dimensional Random Matrices*. Vol. 20. Springer.
- Barabási, Albert-László and Márton Pósfai (2016). *Network Science*. <http://barabasi.com/networksciencebook/>. Cambridge: Cambridge University Press.
- Barrat, Alain et al. (2004). "The Architecture of Complex Weighted Networks". In: *Proceedings of the National Academy of Sciences* 101.11, pp. 3747–3752.
- Bergillos, Carlos (2020). *Ts2vg: Time series to visibility graphs*. <https://pypi.org/project/ts2vg/>. Accessed: 2028-08-09.
- Berndt, Donald J. and James Clifford (1994). "Using Dynamic Time Warping to find Patterns in Time Series". In: *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*. AAAIWS'94. Seattle, WA: AAAI Press, pp. 359–370.
- Bishop, Christopher M (2016). *Pattern Recognition and Machine Learning*. 1st ed. Springer New York, NY.
- Black, Fischer and Myron Scholes (1973). "The Pricing of Options and Corporate Liabilities". In: *Journal of Political Economy* 81.3, pp. 637–654.
- Blondel, Vincent D et al. (2008). "Fast Unfolding of Communities in Large Networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10, P10008.
- Bollerslev, Tim (1986). "Generalized Autoregressive Conditional Heteroskedasticity". In: *Journal of Econometrics* 31.3. <https://www.sciencedirect.com/science/article/pii/0304407686900631>, pp. 307–327. ISSN: 0304-4076. DOI: [https://doi.org/10.1016/0304-4076\(86\)90063-1](https://doi.org/10.1016/0304-4076(86)90063-1).
- Bollerslev, Tim, Ray Y. Chou, and Kenneth F. Kroner (1992). "ARCH Modeling in Finance: A Review of the Theory and Empirical Evidence". In: *Journal of Econometrics* 52.1. <https://www.sciencedirect.com/science/article/pii/030440769290064X>, pp. 5–59. ISSN: 0304-4076. DOI: [https://doi.org/10.1016/0304-4076\(92\)90064-X](https://doi.org/10.1016/0304-4076(92)90064-X).
- Bollerslev, Tim, Robert F Engle, and Daniel B Nelson (1994). "ARCH Models". In: *Handbook of Econometrics* 4, pp. 2959–3038.
- Bonnier, Patric et al. (2019). "Deep Signature Transforms". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.

- Box, George EP et al. (2015). *Time Series Analysis: Forecasting and Control*. 5th ed. John Wiley & Sons.
- Bretto, Alain (2013). *Hypergraph Theory. An Introduction*. Springer International Publishing.
- Brockwell, Peter J and Richard A Davis (1991). *Time Series: Theory and Methods*. Springer Science & Business Media.
- (2002). *Introduction to Time Series and Forecasting*. Springer.
- Buff, Robert (2002). *Uncertain Volatility Models: Theory and Application*. Springer Science & Business Media.
- Candanedo, Luis (2017). *Appliances Energy Prediction*. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5VC8G>.
- Cao, Defu et al. (2020). "Spectral Temporal Graph Neural Network for Multivariate Time-Series Forecasting". In: *Advances in Neural Information Processing Systems* 33, pp. 17766–17778.
- Chen, Chi, Lei Xue, and Wanqi Xing (2023). "Research on Improved GRU-Based Stock Price Prediction Method". In: *Applied Sciences* 13.15, p. 8813.
- Chen, Kuo-Tsai (1958). "Integration of Paths—A Faithful Representation of Paths by Noncommutative Formal Power Series". In: *Transactions of the American Mathematical Society* 89.2. <http://www.jstor.org/stable/1993193>, pp. 395–407. ISSN: 00029947.
- Cheng, Dawei et al. (2022). "Financial time series forecasting with multimodality graph neural network". In: *Pattern Recognition* 121. <https://www.sciencedirect.com/science/article/pii/S00313203210039p>. 108218. ISSN: 0031-3203. DOI: <https://doi.org/10.1016/j.patcog.2021.108218>.
- Chevyrev, Ilya and Andrey Kormilitzin (2016). *A Primer on the Signature Method in Machine Learning*. <https://arxiv.org/abs/1603.03788>. arXiv: 1603.03788 [stat.ML].
- Chevyrev, Ilya and Terry Lyons (2016). "Characteristic Functions of Measures on Geometric Rough Paths". In: *The Annals of Probability* 44.6, pp. 4049–4082.
- Chevyrev, Ilya, Vidit Nanda, and Harald Oberhauser (Jan. 2020). "Persistence Paths and Signature Features in Topological Data Analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.1. <http://dx.doi.org/10.1109/TPAMI.2018.2885516>, pp. 192–202. ISSN: 1939-3539. DOI: 10.1109/tpami.2018.2885516.
- Cho, Kyunghyun et al. (2014). *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. <https://arxiv.org/abs/1409.1259>. arXiv: 1409.1259 [cs.CL].

- Chordia, Tarun, Richard Roll, and Avanidhar Subrahmanyam (2001). "Market Liquidity and Trading Activity". In: *The journal of finance* 56.2, pp. 501–530.
- Chung, Junyoung et al. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. <https://arxiv.org/abs/1412.3555>. arXiv: 1412.3555 [cs.NE].
- Chung, Munki et al. (2022). "The Effects of Errors in Means, Variances, and Correlations on the Mean-Variance Framework". In: *Quantitative Finance* 22.10, pp. 1893–1903.
- Clauset, Aaron, Mark EJ Newman, and Cristopher Moore (2004). "Finding Community Structure in Very Large Networks". In: *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 70.6, p. 066111.
- Clemente, Gian Paolo, Rosanna Grassi, and Asmerilda Hitaj (2021). "Asset Allocation: New Evidence through Network Approaches". In: *Annals of Operations Research* 299.1, pp. 61–80.
- (2022). "Smart Network Based Portfolios". In: *Annals of Operations Research* 316.2, pp. 1519–1541.
- Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2016). *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. <https://arxiv.org/abs/1511.07289>. arXiv: 1511.07289 [cs.LG].
- Cont, Rama (2001). "Empirical Properties of Asset Returns: Stylized Facts and Statistical Issues". In: *Quantitative Finance* 1.2, p. 223.
- Das, Abhimanyu et al. (2024). *Long-term Forecasting with TiDE: Time-series Dense Encoder*. <https://arxiv.org/abs/2304.08424>. arXiv: 2304.08424 [stat.ML].
- De Meer Pardo, Fernando (2019). *Enriching Financial Datasets with Generative Adversarial Networks*.
- De Meer Pardo, Fernando, Peter Schwendner, and Marcus Wunsch (2022). "Tackling the Exponential Scaling of Signature-Based Generative Adversarial Networks for High-Dimensional Financial Time-Series Generation". In: *The Journal of Financial Data Science* 4.4, pp. 110–132.
- Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems* 29.
- Demšar, Janez (2006). "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *The Journal of Machine Learning Research* 7, pp. 1–30.
- Ding, Qianggang et al. (July 2020). "Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction". In: *Proceedings of the*

- Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. <https://doi.org/10.24963/ijcai.2020/640>. International Joint Conferences on Artificial Intelligence Organization, pp. 4640–4646. DOI: 10.24963/ijcai.2020/640.
- Elman, Jeffrey L (1990). “Finding Structure in Time”. In: *Cognitive Science* 14.2, pp. 179–211.
- Engle, Robert F (1982). “Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation”. In: *Econometrica: Journal of the Econometric Society*, pp. 987–1007.
- Engle, Robert F, Sergio M Focardi, and Frank J Fabozzi (2012). “ARCH/GARCH Models in Applied Financial Econometrics”. In: *Encyclopedia of Financial Models*.
- Evertsz, Carl JG (1995). “Fractal Geometry of Financial Time Series”. In: *Fractals* 3.03, pp. 609–616.
- Faloutsos, Christos, M. Ranganathan, and Yannis Manolopoulos (May 1994). “Fast Subsequence Matching in Time-Series Databases”. In: *SIGMOD Rec.* 23.2. <https://doi.org/10.1145/191843.191925>, pp. 419–429. ISSN: 0163-5808. DOI: 10.1145/191843.191925.
- Fama, Eugene F (1970a). “Efficient Capital Markets: A Review of Theory and Empirical Work”. In: *The Journal of Finance* 25.2, pp. 383–417.
- (1970b). “Efficient Capital Markets: A Review of Theory and Empirical Work”. In: *The Journal of Finance* 25.2, pp. 383–417.
- Farida Agustini, W, Ika Restu Affianti, and Endah RM Putri (2018). “Stock Price Prediction using Geometric Brownian Motion”. In: *Journal of Physics: Conference Series*. Vol. 974. IOP Publishing, p. 012047.
- Fasshauer, Gregory E (2007). *Meshfree Approximation Methods with MATLAB*. Singapore: World Scientific.
- Feng, Yifan et al. (2019). “Hypergraph Neural Networks”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33, pp. 3558–3565.
- Fisher, Ronald A (1915). “Frequency Distribution of the Values of the Correlation Coefficient in Samples from an Indefinitely Large Population”. In: *Biometrika* 10.4, pp. 507–521.
- Fortunato, Santo (2010). “Community Detection in Graphs”. In: *Physics reports* 486.3-5, pp. 75–174.
- Francq, Christian and Jean-Michel Zakoian (2019). *GARCH models: Structure, Statistical Inference and Financial Applications*. John Wiley & Sons.
- Friedman, Milton (1937). “The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance”. In: *Journal of the American Statistical Association* 32.200, pp. 675–701.

- Friedman, Milton (1940). "A Comparison of Alternative Tests of Significance for the problem of m Rankings". In: *The Annals of Mathematical Statistics* 11.1, pp. 86–92.
- Fulga, Cristinca, Silvia Dedu, and F Șerban (2009). "Portfolio Optimization with Prior Stock Selection". In: *Economic Computation and Economic Cybernetics Studies and Research* 43.4, pp. 157–172.
- Gao, Yue, Yifan Feng, et al. (2022). "HGNN+: General Hypergraph Neural Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3, pp. 3181–3199.
- Gao, Yue, Meng Wang, et al. (2012). "3-D Object Retrieval and Recognition with Hypergraph Analysis". In: *IEEE transactions on image processing* 21.9, pp. 4290–4303.
- Ghosh, Achyut et al. (2019). "Stock Price Prediction using LSTM on Indian Share Market". In: *Proceedings of the 32nd International Conference on n Computer Applications in Industry and Engineering*. Vol. 63, pp. 101–110.
- Gilmer, Justin et al. (2017). "Neural Message Passing for Quantum Chemistry". In: *International Conference on Machine Learning*. PMLR, pp. 1263–1272.
- Goerg, Georg M (2015). "The Lambert way to Gaussianize Heavy-Tailed Data with the Inverse of Tukey's h Transformation as a Special Case". In: *The Scientific World Journal* 2015.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press.
- Goodfellow, Ian, Jean Pouget-Abadie, et al. (2020). "Generative Adversarial Networks". In: *Communications of the ACM* 63.11, pp. 139–144.
- Graham, Benjamin (2013). *Sparse Arrays of Signatures for Online Character Recognition*. <https://arxiv.org/abs/1308.0371>. arXiv: 1308.0371 [cs.CV].
- Graves, Alex, Santiago Fernández, and Jürgen Schmidhuber (2005). "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition". In: *International Conference on Artificial Neural Networks*. Springer, pp. 799–804.
- Gregnanin, M, J De Smedt, et al. (2024). "Signature-Based Community Detection for Time Series". In: *Complex Networks & Their Applications XII*. Vol. 1142. Cham: Springer Nature Switzerland, pp. 146–158.
- (2025a). "A Generative Adversarial Graph Neural Network for Synthetic Time Series Data". In: *CEUR-WS* 3928.
- (2025b). "Stock Price Time Series Forecasting Using Dynamic Graph Neural Networks and Attention Mechanism in Recurrent Neural Net-

- works". In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Ed. by Rosa Meo and Fabrizio Silvestri. Cham: Springer Nature Switzerland, pp. 357–373. ISBN: 978-3-031-74643-7.
- Gregnanin, M, Y Zhang, et al. (2024). "Signature-based portfolio allocation: a network approach". In: *Applied Network Science* 9. DOI: 10.1007/s41109-024-00651-1.
- Gregnanin, Marco et al. (2026a). *A Generative Adversarial Graph Neural Network for Synthetic Time Series Data*. <https://arxiv.org/abs/2605.22215>. arXiv: 2605.22215 [cs.CE].
- (2026b). *Dynamic Hypergraph Representation Learning for Multivariate Time Series without Prior Knowledge*. <https://arxiv.org/abs/2605.22540>. arXiv: 2605.22540 [cs.CE].
- (2026c). *The Statistical Significance of the Inclusion of Graph Neural Networks in the Financial Time Series Forecasting Problem*. <https://arxiv.org/abs/2605.21192>. arXiv: 2605.21192 [cs.CE].
- Guo, Tian et al. (2016). "Robust Online Time Series Prediction with Recurrent Neural Networks". In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. Ieee, pp. 816–825.
- Gyurkó, Lajos Gergely et al. (2014). *Extracting Information from the Signature of a Financial Data Stream*. <https://arxiv.org/abs/1307.7244>. arXiv: 1307.7244 [q-fin.ST].
- Hamilton, James D (2020). *Time Series Analysis*. Princeton University Press.
- Hamilton, Will, Zhitao Ying, and Jure Leskovec (2017). "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems* 30.
- Heimo, Tapio, Kimmo Kaski, and Jari Saramäki (2009). "Maximal Spanning Trees, Asset Graphs and Random Matrix Denoising in the Analysis of Dynamics of Financial Networks". In: *Physica A: Statistical Mechanics and its Applications* 388.2-3, pp. 145–156.
- Heston, Steven L (1993). "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options". In: *The Review of Financial Studies* 6.2, pp. 327–343.
- Hilpisch, Yves (2015). *Derivatives Analytics with Python: Data Analysis, Models, Simulation, Calibration and Hedging*. John Wiley & Sons.
- Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky (2012). *Neural Networks for Machine Learning: Lecture 6a – Overview of Mini-Batch Gradient Descent*. Technical Report. Dept. of Computer Science, University of Toronto.
- Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780.

- Hua, Yuxiu et al. (2019). "Deep Learning with Long Short-Term Memory for Time Series Prediction". In: *IEEE Communications Magazine* 57.6, pp. 114–119.
- Huang, Jian et al. (2025). "Multimodal Hypergraph Network with Contrastive Learning for Sentiment Analysis". In: *Neurocomputing* 627. <https://www.sciencedirect.com/science/article/pii/S0925231225002383>, p. 129566. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2025.129566>.
- Huang, Jing and Jie Yang (2021). "Unignn: A Unified Framework for Graph and Hypergraph Neural Networks". In: *arXiv preprint arXiv:2105.00956*.
- Huang, Sheng et al. (2015). "Learning Hypergraph-Regularized Attribute Predictors". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 409–417.
- Hyndman, Rob J and Anne B Koehler (2006). "Another Look at Measures of Forecast Accuracy". In: *International journal of forecasting* 22.4, pp. 679–688.
- Jiang, Jianwen et al. (2019). "Dynamic Hypergraph Neural Networks." In: *28th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2635–2641.
- Jin, Taisong et al. (2019). "Robust l2-Hypergraph and its Applications". In: *Information Sciences* 501, pp. 708–723.
- Jing, Ruixue and Luis EC Rocha (2023). "A Network-based Strategy of Price Correlations for Optimal Cryptocurrency Portfolios". In: *Finance Research Letters* 58, p. 104503.
- Kalsi, Jasdeep, Terry Lyons, and Imanol Perez Arribas (2020). "Optimal Execution with Rough Path Signatures". In: *SIAM Journal on Financial Mathematics* 11.2. <https://doi.org/10.1137/19M1259778>, pp. 470–493. DOI: 10.1137/19M1259778. eprint: <https://doi.org/10.1137/19M1259778>.
- Kang, Xiaojun et al. (2022). "Dynamic Hypergraph Neural Networks Based On Key Hyperedges". In: *Information Sciences* 616, pp. 37–51.
- Kidger, Patrick and Terry Lyons (2020). *Signatory: Differentiable Computations of the Signature and Logsignature Transforms, on both CPU and GPU*. <https://arxiv.org/abs/2001.00706>. arXiv: 2001.00706 [cs.LG].
- Kidger, Patrick, James Morrill, et al. (2020). "Neural Controlled Differential Equations for Irregular Time Series". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20*. Vancouver, BC, Canada: Curran Associates Inc. ISBN: 9781713829546.

- Kingma, Diederik P and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980*.
- Kipf, Thomas N. and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations*. <https://openreview.net/forum?id=SJU4ayYgl>.
- Kolm, Petter N, Reha Tütüncü, and Frank J Fabozzi (2014). "60 Years of Portfolio Optimization: Practical Challenges and Current Trends". In: *European Journal of Operational Research* 234.2, pp. 356–371.
- Kondratyev, Alexei and Christian Schwarz (2019). *The Market Generator*.
- Kormilitzin, A. B. et al. (2016). *Application of the Signature Method to Pattern Recognition in the CEQUEL Clinical Trial*. <https://arxiv.org/abs/1606.02074>. arXiv: 1606.02074 [stat.AP].
- Kormilitzin, Andrey et al. (2017). *Detecting Early Signs of Depressive and Manic Episodes in Patients with Bipolar Disorder using the Signature-Based Model*. <https://arxiv.org/abs/1708.01206>. arXiv: 1708.01206 [stat.ML].
- Kou, Steven G (2002). "A Jump-Diffusion Model for Option Pricing". In: *Management Science* 48.8, pp. 1086–1101.
- Kullback, Solomon and Richard A Leibler (1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86.
- Lacasa, Lucas et al. (2008). "From Time Series to Complex Networks: The Visibility Graph". In: *Proceedings of the National Academy of Sciences* 105.13, pp. 4972–4975.
- Laloux, Laurent et al. (1999). "Noise Dressing of Financial Correlation Matrices". In: *Physical review letters* 83.7, p. 1467.
- Lamberton, Damien and Bernard Lapeyre (2011). *Introduction to Stochastic Calculus Applied to Finance*. CRC press.
- Lantao, Yu et al. (2017). "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient". In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, pp. 2852–2858.
- Lazcano, Ana, Pedro Javier Herrera, and Manuel Monge (2023). "A Combined Model Based on Recurrent Neural Networks and Graph Convolutional Networks for Financial Time Series Forecasting". In: *Mathematics* 11.1, p. 224.
- Lea, Colin et al. (2016). *Temporal Convolutional Networks: A Unified Approach to Action Segmentation*. <https://arxiv.org/abs/1608.08242>. arXiv: 1608.08242 [cs.CV].

- Lee, San K, LT Nguyen, and Malick O Sy (2017). "Comparative Study of Volatility Forecasting Models: The Case of Malaysia, Indonesia, Hong Kong and Japan stock markets". In: *Economics* 5.4, pp. 299–310.
- Lemercier, Maud et al. (2021). "Distribution Regression for Sequential Data". In: *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research, pp. 3754–3762.
- Leung, Mark T, Hazem Daouk, and An-Sing Chen (2000). "Forecasting Stock Indices: A Comparison of Classification and Level Estimation Models". In: *International Journal of forecasting* 16.2, pp. 173–190.
- Levin, D., T. Lyons, and H. Ni (2016). *Learning from the Past, Predicting the Statistics for the Future, Learning an Evolving System*. <https://arxiv.org/abs/1309.0260>. arXiv: 1309.0260 [q-fin.ST].
- Levin, Daniel, Terry Lyons, and Hao Ni (2013). *Learning from the Past, Predicting the Statistics for the Future, Learning an Evolving System*. arXiv: 1309.0260 [q-fin.ST].
- Li, Lei and Tao Li (2013). "News Recommendation Via Hypergraph Learning: Encapsulation of User Behavior and News Content". In: *Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 305–314.
- Li, Shu et al. (2025). "Prototype-Oriented Hypergraph Representation Learning for Anomaly Detection in Tabular Data". In: *Information Processing & Management* 62.1. <https://www.sciencedirect.com/science/article/pii/S030645732400236X>, p. 103877. ISSN: 0306-4573. DOI: <https://doi.org/10.1016/j.ipm.2024.103877>.
- Li, Xi et al. (Oct. 2014). "Context-Aware Hypergraph Construction for Robust Spectral Clustering". In: *IEEE Transactions on Knowledge and Data Engineering* 26.10. <http://dx.doi.org/10.1109/TKDE.2013.126>, pp. 2588–2597. ISSN: 1041-4347. DOI: 10.1109/tkde.2013.126.
- Li, Yan et al. (2019). "Portfolio Optimization Based on Network Topology". In: *Physica A: Statistical Mechanics and its Applications* 515, pp. 671–681.
- Liao, S. et al. (2023). *Conditional Sig-Wasserstein GANs for Time Series Generation*. <https://arxiv.org/abs/2006.05421>. arXiv: 2006.05421 [cs.LG].

- Lin, Yuyang et al. (2022). "A New Attention-Based LSTM Model for Closing Stock Price Prediction". In: *International Journal of Financial Engineering* 9.03, p. 2250014.
- Lu, Wenjie et al. (2021). "A CNN-BiLSTM-AM Method for Stock Price Prediction". In: *Neural Computing and Applications* 33, pp. 4741–4753.
- Luo, Xiaoyi, Jiaheng Peng, and Jun Liang (2022). "Directed Hypergraph Attention Network for Traffic Forecasting". In: *IET Intelligent Transport Systems* 16.1, pp. 85–98.
- Lyons, Terry (2014). *Rough Paths, Signatures and the Modelling of Functions on Streams*. <https://arxiv.org/abs/1405.4537>. arXiv: 1405.4537 [math.PR].
- Lyons, Terry, Sina Nejad, and Imanol Perez Arribas (2020). "Non-parametric Pricing and Hedging of Exotic Derivatives". In: *Applied Mathematical Finance* 27.6. <https://doi.org/10.1080/1350486X.2021.1891555>, pp. 457–494. DOI: 10.1080/1350486X.2021.1891555. eprint: <https://doi.org/10.1080/1350486X.2021.1891555>.
- Lyons, Terry and Hao Ni (2015). "Expected Signature of Brownian Motion up to the First Exit Time from a Bounded Domain". In: *The Annals of Probability* 43.5, pp. 2729–2762.
- Lyons, Terry, Hao Ni, and Harald Oberhauser (2014). "A feature set for streams and an application to high-frequency financial tick data". In: *Proceedings of the 2014 International Conference on Big Data Science and Computing*, pp. 1–8.
- Lyons, Terry J (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310.
- Ma, Zitong, Wenbo Zhao, and Zhe Yang (2024). "Directed Hypergraph Representation Learning for Link Prediction". In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. Ed. by Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li. Vol. 238. *Proceedings of Machine Learning Research*. <https://proceedings.mlr.press/v238/ma24b.html>. PMLR, pp. 3268–3276.
- MacMahon, Mel and Diego Garlaschelli (Apr. 2015). "Community Detection for Correlation Matrices". In: *Physical Review X* 5.2. ISSN: 2160-3308.
- Mandelbrot, Benoit B (2013). *Fractals and Scaling in Finance: Discontinuity, Concentration, Risk. Selecta volume E*. Springer Science & Business Media.
- Mansini, Renata, Włodzimierz Ogryczak, and M Grazia Speranza (2014). "Twenty Years of Linear Programming based Portfolio Optimization". In: *European Journal of Operational Research* 234.2, pp. 518–535.

- Mantegna, Rosario N and H Eugene Stanley (1999). *Introduction to Econophysics: Correlations and Complexity in Finance*. Cambridge: Cambridge University Press.
- Markowitz, Harry (1952). "Portfolio Selection". In: *The Journal of Finance* 7.1. <http://www.jstor.org/stable/2975974>, pp. 77–91. ISSN: 00221082, 15406261.
- Mba, Jules Clément, Sutene Mwambetania Mwambi, and Edson Pindza (2022). "A Monte Carlo Approach to Bitcoin Price Prediction with Fractional Ornstein-Uhlenbeck Levy Process". In: *Forecasting* 4.2, pp. 409–419.
- Mehta, Madan Lal (2004). *Random Matrices*. Vol. 142. Elsevier.
- Merton, Robert C (1974). "On the Pricing of Corporate Debt: The risk Structure of Interest Rates". In: *The Journal of Finance* 29.2, pp. 449–470.
- (1976). "Option Pricing when Underlying Stock Returns are Discontinuous". In: *Journal of Financial Economics* 3.1-2, pp. 125–144.
- Meucci, Attilio (2005). *Risk and Asset Allocation*. Vol. 1. Springer.
- Mogren, Olof (2016). "C-RNN-GAN: A Continuous Recurrent Neural Network with Adversarial Training". In: *Constructive Machine Learning Workshop (CML) at NIPS 2016*, p. 1.
- Moore, P. J., T. J. Lyons, and J. Gallacher (Sept. 2019). "Using Path Signatures to Predict a Diagnosis of Alzheimer's Disease". In: *PLOS ONE* 14.9. Ed. by Stephen D Ginsberg. <http://dx.doi.org/10.1371/journal.pone.0222212>, e0222212. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0222212.
- Moreno-Pino, Fernando et al. (2024). "Rough Transformers: Lightweight and Continuous Time Series Modelling through Signature Patching". In: *Proceedings of the 38th International Conference on Neural Information Processing Systems*. NIPS '24. Vancouver, BC, Canada: Curran Associates Inc. ISBN: 9798331314385.
- Morrill, James et al. (2021). *A Generalised Signature Method for Multivariate Time Series Feature Extraction*. <https://arxiv.org/abs/2006.00873>. arXiv: 2006.00873 [cs.LG].
- Nemenyi, Peter Bjorn (1963). *Distribution-Free Multiple Comparisons*. Princeton University.
- Newman, Mark EJ and Michelle Girvan (2004). "Finding and Evaluating Community Structure in Networks". In: *Physical review E* 69.2, p. 026113.
- Ni, Hao et al. (2020). *Conditional Sig-Wasserstein GANs for Time Series Generation*. arXiv: 2006.05421 [cs.LG].

- Nie, Yuqi et al. (2023). "A Time Series is Worth 64 Words: Long-term Forecasting with Transformers". In: *International Conference on Learning Representations*.
- Onnela, J-P, Kimmo Kaski, and Janos Kertész (2004). "Clustering and Information in Correlation Based Financial Networks". In: *The European Physical Journal B* 38.2, pp. 353–362.
- Onnela, Jukka-Pekka et al. (June 2005). "Intensity and Coherence of Motifs in Weighted Complex Networks". In: *Physical Review E* 71.6. ISSN: 1550-2376.
- Oreshkin, Boris N. et al. (2020). "N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting". In: *International Conference on Learning Representations*. <https://openreview.net/forum?id=rlecqn4YwB>.
- Peralta, Gustavo and Abalfazl Zareei (2016). "A Network Approach to Portfolio Selection". In: *Journal of Empirical Finance* 38, pp. 157–180.
- Peters, Edgar E (1994). *Fractal Market Analysis: Applying Chaos Theory to Investment and Economics*. Vol. 24. John Wiley & Sons.
- Peterson, Leif E (2009). "K-Nearest Neighbor". In: *Scholarpedia* 4.2, p. 1883.
- Plerou, Vasiliki, Parameswaran Gopikrishnan, Bernd Rosenow, Luís A Nunes Amaral, Thomas Guhr, et al. (2002). "Random Matrix Approach to Cross Correlations in Financial Data". In: *Physical Review E* 65.6, p. 066126.
- Plerou, Vasiliki, Parameswaran Gopikrishnan, Bernd Rosenow, Luís A Nunes Amaral, and H Eugene Stanley (1999). "Universal and Nonuniversal Properties of Cross Correlations in Financial Time Series". In: *Physical review letters* 83.7, p. 1471.
- Potters, M., J. P. Bouchaud, and L. Laloux (2005). *Financial Applications of Random Matrix Theory: Old Laces and New Pieces*. <https://arxiv.org/abs/physics/0507111>. arXiv:physics/0507111 [physics.data-an-
- Prigent, Jean-Luc (2007). *Portfolio Optimization and Performance Analysis*. Chapman and Hall/CRC.
- Rehman, Rana Zia Ur et al. (2020). "Gait Analysis with Wearables Can Accurately Classify Fallers from Non-Fallers: A Step toward Better Management of Neurological Disorders". In: *Sensors* 20.23. <https://www.mdpi.com/1424-8220/20/23/6992>. ISSN: 1424-8220. DOI: 10.3390/s20236992.
- Resnick, Sidney (2019). *A Probability Path*. Springer.
- Ricca, Federica and Andrea Scozzari (2024). "Portfolio Optimization through a Network Approach: Network Assortative Mixing and Portfolio Di-

- versification". In: *European Journal of Operational Research* 312.2, pp. 700–717. ISSN: 0377-2217.
- Roondiwala, Murtaza, Harshal Patel, and Shraddha Varma (2017). "Predicting Stock Prices using LSTM". In: *International Journal of Science and Research (IJSR)* 6.4, pp. 1754–1756.
- Rounaghi, Mohammad Mahdi and Farzaneh Nassir Zadeh (2016). "Investigation of Market Efficiency and Financial Stability Between S&P 500 and London Stock Exchange: Monthly and Yearly Forecasting of Time Series Stock Returns using ARMA Model". In: *Physica A: Statistical Mechanics and its Applications* 456, pp. 10–21.
- Rubner, Yossi, Carlo Tomasi, and Leonidas J Guibas (2000). "The Earth Mover's Distance as a Metric for Image Retrieval". In: *International Journal of Computer Vision* 40, pp. 99–121.
- Ruiz, Esther and Lorenzo Pascual (2002). "Bootstrapping Financial Time Series". In: *Journal of Economic Surveys* 16.3, pp. 271–300.
- Ruiz, Luana, Fernando Gama, and Alejandro Ribeiro (2020). "Gated Graph Recurrent Neural Networks". In: *IEEE Transactions on Signal Processing* 68, pp. 6303–6318.
- Salzberg, Steven L (1997). "On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach". In: *Data Mining and Knowledge Discovery* 1, pp. 317–328.
- Sandhu, Romeil S., Tryphon T. Georgiou, and Allen R. Tannenbaum (2016). "Ricci Curvature: An Economic Indicator for Market Fragility and Systemic Risk". In: *Science Advances* 2.5. <https://www.science.org/doi/abs/10.1126/sciadv.1501495>, e1501495. DOI: 10.1126/sciadv.1501495. eprint: <https://www.science.org/doi/pdf/10.1126/sciadv.1501495>.
- Sawhney, Ramit et al. (2021). "Stock Selection via Spatiotemporal Hypergraph Attention Network: A Learning to Rank Approach". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.1. <https://ojs.aaai.org/index.php/AAAI/article/view/16127>, pp. 497–504. DOI: 10.1609/aaai.v35i1.16127.
- Scarselli, Franco et al. (2008). "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1, pp. 61–80.
- Schuster, Mike and Kuldip K Paliwal (1997). "Bidirectional Recurrent Neural Networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681.
- Selvin, Sreelekshmy et al. (2017). "Stock Price Prediction using LSTM, RNN and CNN-Sliding Window Model". In: *2017 International Con-*

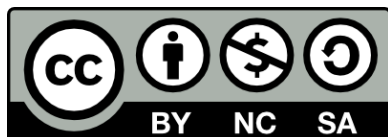
- ference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, pp. 1643–1647.
- Sezer, Omer Berat, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu (2020). “Financial Time Series Forecasting with Deep Learning: A Systematic Literature Review: 2005–2019”. In: *Applied soft computing* 90, p. 106181.
- Sharpe, William F (1998). “The Sharpe Ratio”. In: *Streetwise—the Best of the Journal of Portfolio Management* 3, pp. 169–185.
- Sheskin, David J (2003). *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman and hall/CRC.
- Sinha, Sitabhra et al. (2010). *Econophysics: an Introduction*. John Wiley & Sons.
- Skarding, Joakim, Bogdan Gabrys, and Katarzyna Musial (2021). “Foundations and Modeling of Dynamic Networks using Dynamic Graph Neural Networks: A Survey”. In: *IEEE Access* 9, pp. 79143–79168.
- Stephen, Mutua, Changgui Gu, and Huijie Yang (2015). “Visibility Graph Based Time Series Analysis”. In: *PloS One* 10.11, e0143015.
- Sun, He et al. (2023). *Decision-Aware Conditional GANs for Time Series Data*. arXiv: 2009.12682 [cs.LG].
- Sun, Xiangguo et al. (2023). “Self-Supervised Hypergraph Representation Learning for Sociological Analysis”. In: *IEEE Transactions on Knowledge and Data Engineering* 35.11, pp. 11860–11871. DOI: 10.1109/TKDE.2023.3235312.
- Tankov, Peter and Rama Cont (2003). *Financial Modelling with Jump Processes*. CRC Press.
- Tong, Anh et al. (2023). “SigFormer: Signature Transformers for Deep Hedging”. In: *Proceedings of the Fourth ACM International Conference on AI in Finance*. ICAIF ’23. <https://doi.org/10.1145/3604237.3626841>. Brooklyn, NY, USA: Association for Computing Machinery, pp. 124–132. ISBN: 9798400702402. DOI: 10.1145/3604237.3626841.
- Tsay, Ruey S (2005). *Analysis of Financial Time Series*. John Wiley & Sons.
- Tumminello, Michele et al. (2005). “A Tool for Filtering Information in Complex Systems”. In: *Proceedings of the National Academy of Sciences* 102.30, pp. 10421–10426.
- Utsugi, Akihiko, Kazusumi Ino, and Masaki Oshikawa (2004). “Random Matrix Theory Analysis of Cross Correlations in Financial Markets”. In: *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 70.2, p. 026110.

- Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems* 30.
- Veličković, Petar (2022). "Message Passing All the Way Up". In: *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*. <https://openreview.net/forum?id=Bc8GiEZkTe5>.
- Veličković, Petar et al. (2018). "Graph Attention Networks". In: *International Conference on Learning Representations*. <https://openreview.net/forum?id=rJXMpikCZ>.
- Vidović, Jelena (2019). "Turnover Based Illiquidity Measurement as Investment Strategy on Zagreb Stock Exchange". In: *American Journal of Operations Research* 10.1, pp. 1–12.
- Villani, Cédric et al. (2009). *Optimal Transport: Old and New*. Vol. 338. Springer.
- Vito, Saverio (2008). *Air Quality*. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C59K5F>.
- Výrost, Tomas, Štefan Lyócsa, and Eduard Baumöhl (2019). "Network-based Asset Allocation Strategies". In: *The North American Journal of Economics and Finance* 47, pp. 516–536.
- Wang, Hui (2012). *Monte Carlo Simulation with Applications to Finance*. CRC Press.
- Wang, Jingcheng et al. (2022). "Multitask Hypergraph Convolutional Networks: A Heterogeneous Traffic Prediction Framework". In: *IEEE Transactions on Intelligent Transportation Systems* 23.10, pp. 18557–18567.
- Wang, Shiyu et al. (2024). "TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting". In: *International Conference on Learning Representations (ICLR)*.
- Wang, Shun et al. (2024). "Dynamic Hypergraph Structure Learning for Multivariate Time Series Forecasting". In: *IEEE Transactions on Big Data*.
- Wang, Yu-Tian et al. (2021). "MiRNA-Disease Association Prediction via Hypergraph Learning Based on High-Dimensionality Features". In: *BMC Medical Informatics and Decision Making* 21.Suppl 1, p. 133.
- Wang, Yishu et al. (2019). "Time-Dependent Graphs: Definitions, Applications, and Algorithms". In: *Data Science and Engineering* 4.4, pp. 352–366.
- Wei, Ze-Lin et al. (2025). "FSTGAT: Financial Spatio-Temporal Graph Attention Network for Non-Stationary Financial Systems and Its Application in Stock Price Prediction". In: *Symmetry* 17.8. <https://www.mdpi.com/2073-8994/17/8/1344>. ISSN: 2073-8994. DOI: 10.3390/sym17081344.

- Welch, Ivo and Amit Goyal (2008). "A Comprehensive Look at the Empirical Performance of Equity Premium prediction". In: *The Review of Financial Studies* 21.4, pp. 1455–1508.
- Wiese, Magnus et al. (2020). "Quant GANs: Deep Generation of Financial Time Series". In: *Quantitative Finance* 20.9, pp. 1419–1440.
- Wilcoxon, Frank (1992). "Individual Comparisons by Ranking Methods". In: *Breakthroughs in Statistics: Methodology and Distribution*. Springer, pp. 196–202.
- Wu, Haixu et al. (2023). *TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis*. <https://arxiv.org/abs/2210.02186>. arXiv: 2210.02186 [cs.LG].
- Wu, Zonghan, Shirui Pan, Fengwen Chen, et al. (2020). "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1, pp. 4–24.
- Wu, Zonghan, Shirui Pan, Guodong Long, et al. (2019). *Graph WaveNet for Deep Spatial-Temporal Graph Modeling*. <https://arxiv.org/abs/1906.00121>. arXiv: 1906.00121 [cs.LG].
- Xia, Hongjie et al. (2024). "CI-STHPAN: Pre-Trained Attention Network for Stock Selection with Channel-Independent Spatio-Temporal Hypergraph". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.8. <https://ojs.aaai.org/index.php/AAAI/article/view/28770>, pp. 9187–9195. DOI: 10.1609/aaai.v38i8.28770.
- Xiang, Sheng et al. (2022). "Temporal and Heterogeneous Graph Neural Network for Financial Time Series Prediction". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. CIKM '22. <https://doi.org/10.1145/3511808.3557089>. Atlanta, GA, USA: Association for Computing Machinery, pp. 3584–3593. ISBN: 9781450392365. DOI: 10.1145/3511808.3557089.
- Xiong, Caiming, Stephen Merity, and Richard Socher (2016). "Dynamic Memory Networks for Visual and Textual Question Answering". In: *International Conference on Machine Learning*. PMLR, pp. 2397–2406.
- Xu, Keyulu, Weihua Hu, et al. (2019). "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. <https://openreview.net/forum?id=ryGs6iA5Km>.
- Xu, Keyulu, Chengtao Li, et al. (2018). "Representation Learning on Graphs with Jumping Knowledge Networks". In: *International Conference on Machine Learning*. PMLR, pp. 5453–5462.

- Yang, Weixin et al. (2022). "Developing the Path Signature Methodology and Its Application to Landmark-Based Human Action Recognition". In: *Stochastic Analysis, Filtering, and Stochastic Optimization: A Commemorative Volume to Honor Mark H. A. Davis's Contributions*. https://doi.org/10.1007/978-3-030-98519-6_18. Cham: Springer International Publishing, pp. 431–464. ISBN: 978-3-030-98519-6. DOI: 10.1007/978-3-030-98519-6_18.
- Ye, Lexiang and Eamonn Keogh (2011). "Time Series Shapelets: A Novel Technique that Allows Accurate, Interpretable and Fast Classification". In: *Data Mining and Knowledge Discovery* 22. <https://doi.org/10.1007/s10618-010-0179-5>, pp. 149–182. DOI: 10.1007/s10618-010-0179-5.
- Yi, Jaehyuk and Jinkyoo Park (2020). "Hypergraph Convolutional Recurrent Neural Network". In: *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3366–3376.
- Yin, Nan, Fuli Feng, et al. (2022). "Dynamic Hypergraph Convolutional Network". In: *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, pp. 1621–1634.
- Yin, Nan, Li Shen, et al. (2024). "Messages are Never Propagated Alone: Collaborative Hypergraph Neural Network for Time-Series Forecasting". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.4, pp. 2333–2347. DOI: 10.1109/TPAMI.2023.3331389.
- Yoon, Jinsung, Daniel Jarrett, and Mihaela van der Schaar (2019). "Time-series Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc.
- You, Jiaxuan, Tianyu Du, and Jure Leskovec (2022). "ROLAND: Graph learning Framework for Dynamic Graphs". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2358–2366.
- Yu, Bing, Haoteng Yin, and Zhanxing Zhu (July 2018). "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. IJCAI-2018. <http://dx.doi.org/10.24963/ijcai.2018/505>. International Joint Conferences on Artificial Intelligence Organization. DOI: 10.24963/ijcai.2018/505.
- Zhang, Aston et al. (2023). *Dive into Deep Learning*. <https://D2L.ai>. Cambridge University Press.

- Zhang, Liheng, Charu Aggarwal, and Guo-Jun Qi (2017). "Stock Price Prediction via Discovering Multi-Frequency Trading Patterns". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2141–2149.
- Zhang, Zizhao et al. (2018). "Dynamic Hypergraph Structure Learning." In: *27th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3162–3169.
- Zhao, Geya, Minggao Xue, and Li Cheng (2023). "A new hybrid model for multi-step WTI futures price forecasting based on self-attention mechanism and spatial-temporal graph neural network". In: *Resources Policy* 85. <https://www.sciencedirect.com/science/article/pii/S0301420723006670>, p. 103956. ISSN: 0301-4207. DOI: <https://doi.org/10.1016/j.resourpol.2023.103956>.
- Zhao, Ling et al. (2019). "T-GCN: A temporal Graph Convolutional Network for Traffic Prediction". In: *IEEE Transactions on Intelligent Transportation Systems* 21.9, pp. 3848–3858.



Unless otherwise expressly stated, all original material of whatever nature created by Marco Gregnanin and included in this thesis, is licensed under a Creative Commons Attribution Noncommercial Share Alike 3.0 Italy License.

Check on Creative Commons site:

<https://creativecommons.org/licenses/by-nc-sa/3.0/it/legalcode/>

<https://creativecommons.org/licenses/by-nc-sa/3.0/it/deed.en>

Ask the author about other uses.